

KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Mechanical Engineering

Probabilistic Human-Robot Navigation

Plan Recognition, User Modelling and Shared Control for Robotic Wheelchairs

Alexander HÜNTEMANN

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

2011D03

March 2011



KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Mechanical Engineering

Probabilistic Human-Robot Navigation

Plan Recognition, User Modelling and Shared Control for Robotic Wheelchairs

Alexander HÜNTEMANN

Jury:
Prof. dr. Adhemar Bultheel, (chair)
Prof. dr. ir. Hendrik Van Brussel, (promotor)
Prof. dr. ir. Marnix Nuttin, (promotor)
Prof. dr. ir. Joris De Schutter
Prof. dr. ir. Johan Suykens
Prof. dr. ir. Herman Bruyninckx
Prof. dr. Ann Nowé
(Vrije Universiteit Brussel, Belgium)
Prof. Dr. Wolfram Burgard
(Albert-Ludwigs-Universität Freiburg, Germany)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

2011D03

March 2011

© Katholieke Universiteit Leuven – Faculty of Engineering
Department of Mechanical Engineering, Celestijnenlaan 300B, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2011/7515/27
ISBN 978-94-6018-325-6

A mi abuela María

Preface

"None of my inventions came by accident. I see a worthwhile need to be met and I make trial after trial until it comes. What it boils down to is one per cent inspiration and ninety-nine per cent perspiration."

Thomas Edison ($\star$1847 – †1931)
Statement in a press conference (1929) as quoted by (Newton, 1989)

Dear reader,

Thank you for your time and curiosity. Even though I do not know who you are, I have thought frequently about you while writing this thesis. You are now holding in your hands a peculiar mix, 99% perspiration and 1% inspiration according to Edison. Research could not have been described more accurately. The rest of the book reflects the results of research perspiration. Hopefully, it does not require excessive effort from your side. In this section, which stretches over roughly 1% of the book, I would like to introduce you to my sources of inspiration and also acknowledge the people who sweated together with me to make this work possible. I am deeply indebted to them because this work cannot be understood without their crucial contributions.

First, I would like to thank my promotor Hendrik Van Brussel for his guidance throughout the research. His innate curiosity has been very inspiring and his insights have shaped my understanding of human-robot navigation. Rik, I wish you all the best for your retirement. Thanks also to my co-promotor Marnix Nuttin, who welcomed me in the Mobile Learning Robots (MLR) research group and helped to get the research started. Additionally, I wish to thank Eric Demeester who is not a formal promotor, but has mentored me from the start. Eric, fellow Bayesian roboticist, thanks for the fruitful collaboration, the guidance, all interesting discussions and for reading parts of the text.

I am very grateful to all members of the jury Prof. Joris De Schutter, Prof. Johan Suykens, Prof. Ann Nowé, Prof. Herman Bruyninckx, Prof. Wolfram Burgard and Prof. Adhemar Bultheel. Their invaluable comments and suggestions have improved the text significantly. Prof. Burgard, herzlichen Dank, dass Sie

die weite Reise unternommen haben, um an der Beurteilung der Dissertation teilzunehmen. Die Diskussion über die probabilistische Interpretation der Pfad-Metrik hat mir zu einem neuen Verständnis der probabilistischen Struktur meines lokalen Planerkennungs-Ansatzes verholfen.

The goal of this thesis is to help the elderly and disabled to regain their independence through robotised wheelchairs. Therefore, the navigation experiments[1] with a real wheelchair user of chapter 4 are crucial to this work. These experiments were performed in the course of the EU-IST MOVEMENT project. I am very grateful to all members of the MOVEMENT consortium for the productive collaboration, which made these experiments possible. Thanks Wolfgang Zagler, Peter Mayer, Georg Edelmayer, Markus Vincze, Peter Einramhof, Andreas Hochgatterer, Thomas Fuxreiter, Gert Jan Gelderblom, Ger Cremers, Nicola Tomatis, Ralph Piguet, Remy Blank, Thomas Brendel, Andrea Pisetta, Ulrich Unterhinninghofen, Georgios Lidoris, Gernot Kronreif, and Martin Fürst. Special thanks to the staff at iRv (Ger Cremers and Gert Jan Gelderblom) for the assistance during the actual navigation experiments. I am also deeply thankful to the users, who were eager to test the robotic wheelchair and to contribute with their experience to its development. These users are remarkable people. Their courage, motivation and desire to overcome their limitations is admirable and has inspired me greatly. I also appreciate the effort and dedication of the family members and therapists, who accompanied the users to the experiments and explained me all the difficulties disabled people normally face.

The research of this thesis was made possible through the financial support of the European Union, through the projects MOVEMENT and RADHAR, and by the Research Foundation - Flanders (FWO), which I gratefully acknowledge. Thanks, promotors, for ensuring these sources of financing. I am thankful to Herman Bruyninckx for bridging the gap between the end of my research fellowship and the start of the RADHAR project.

The newly started RADHAR project will extend the research described in this thesis to more challenging environments, and to a larger user population interacting with the robotic wheelchair over long periods of time. RADHAR (*R*obotic *AD*aptation to *H*umans *A*dapting to *R*obots) will study the life-long learning aspect of human-robot navigation in dynamic environments. It was fun, but also much perspiration, to write the RADHAR proposal with Manu and Eric through many days and nights. Thanks also to all partners of RADHAR who accepted the research challenge.

Working at K.U.Leuven has been a very enriching experience on the professional as well as on the personal level. Thanks to all the members of MLR, Eric, Dirk, Manu, Johan, Gerolf, Alexandra, Shu, Davy, Xavier, Brecht, Hsiao-Wei, and Erwin. Working together with you has been a pleasure and your friendship is warmly appreciated. Special thanks also to my office mates, Johan, Gerolf, Shu and Alexandra. The atmosphere in the office has always been great. Thanks also to

[1] special effect: dear reader, you can actually see the wheelchair moving if you browse very quickly through the pages of section 4.3.4.

all my new colleagues in the Robotics group (Ruben, Nick, Koen, Enrico, Steven, Wilm, Tinne, Hugo, Markus, Hans, Lin, Joris and Herman) for their warm welcome when MLR joined their group last year. Thanks, Tinne, for the shared learning experiences at several Summer Schools, the interesting discussions about Bayesian probability and for the deduction of equation 3.28. Thanks also to all technical and administrative staff of the department for their kind support. Outside of the research group but still at K.U.Leuven, I have met wonderful people as well: Antonia, Bart, Cinzia, Clara, Cristobal, Donato, Denys, Eleonora, Friedl, Gorka, Inge, Jan, Kun, Leopoldo, Maira, Mauro, Mohamed, Natalia, Silvia, Stella, Thierry, and Thomas. Clara, thanks for the beautiful drawing on the back-cover of the thesis.

Beyond the confines of the lab, I am grateful to the friends in Leuven with whom I have spent such great times. Thank you Adriana, Angels, Leti, Laura, Marilu, Raquel, Lorenzo, Naim, Sara, Karen, Dimitri, and Patricia. A mis amigos en Madrid, también os quiero agradecer tantos momentos memorables que hemos compartido juntos. Gracias Mario, Ana, Edu, Lore, José L. R., Leti, Andrés, José L. S., Miguel, Carlos, Antonio, Rubén, Laura, José A., Bea, Javi, Inés, Mariola, e Isabel. Isa, muchas gracias por acompañarme parte del camino.

Finalmente le quiero agradecer a mi familia de todo corazón el apoyo y cariño que me han mostrado. Muchas gracias por confiar en mí y por apoyarme en los momentos difíciles. Gracias mamá, vielen Dank Papa, gracias Katia, Fausto y a mi abuela María. Julie, bedankt voor je liefde en voor je steun.

Before concluding, I want to add a comment on language. This thesis is dedicated to the memory of my grandmother. In her honour and in order to make the reading more fluent, the user throughout this thesis is assumed female. Please note, however, that the findings of this work are not gender-specific.

Dear reader, I hope you find some answers about human-robot navigation in the remainder of the text or that it inspires you new research questions. I wish you a pleasant reading,

Alexander Hüntemann
Leuven, March 2011

Abstract

Challenge. The demographics of society are changing because of an ageing population. As a consequence, disabilities related to ageing are becoming more prevalent. Further, fewer young people are available to attend the elderly or disabled. Such evolution is problematic since disabilities often involve the loss of mobility and affected individuals depend on others for help. Impaired mobility can also lead to isolation. Both effects combined potentially reduce a person's well-being and self-esteem. Besides the important personal aspect, loss of mobility and dependency can also have a grave economic impact on society as a whole.

Fact. Mobility enhancing technology, such as electrical wheelchairs, addresses the demographic challenge by helping people to regain their independence. However, electrical wheelchairs are difficult to control and dangerous accidents with the environment and other people are rife. Therefore, many potential users are not prescribed an electrical wheelchair if they lack the necessary cognitive or physical skills. Even able users find it difficult and tiresome to navigate a wheelchair safely.

Opportunity. The past decade has witnessed tremendous advances in mobile robotics. Autonomous cars were racing through the dessert[2] or navigating in a city[3]. Mobile robotics is also finding its way to mainstream products, such as vacuum cleaners or lawn mowers[4]. Hence, mobile robotics is an ideal technology for helping the elderly and disabled to navigate safely in almost any environment.

Challenge. The elderly and disabled are a very heterogeneous user group, with different abilities and needs. The challenge is to develop *customised robots*, which can *work together effectively* with such a *heterogeneous population*, but still remain *easy* and *intuitive* to control.

Requirements. Such a heterogeneous user group requires a general methodology that can adapt navigation assistance to each individual. In order to avoid frustrating its user, the robot should attempt to recognise its user's navigation plan in a way that considers the user's driving abilities. Furthermore, the robot should provide assistance only when the user requires it. Users should be encouraged to participate in navigation as far as possible to remain mentally and physically fit.

[2] the DARPA Grand Challenge (Buehler et al., 2007)
[3] the DARPA Urban Grand Challenge (Buehler et al., 2009)
[4] Roomba vacuum cleaner (iRobot, 2011) and lawn mowers (Robomow, 2011; Husqvarna, 2011)

Methodology. In this thesis we propose a *probabilistic* framework to *recognise the user's navigation plans* out of a set of *local trajectories*. In order to increase robustness, this framework considers the uncertainty when recognising user plans. It fuses past driving information with the user's specific driving style in order to estimate a posterior probability over user plans. Further, these estimated plans are the basis to share the control over the wheelchair according to the user's abilities and needs. The robot ensures the user's safety because it reasons about collision-free trajectories that are also physically executable, i.e. kinematically and dynamically feasible.

The first contribution of this thesis is a probabilistic formulation of human-robot navigation as a Dynamic Bayesian Network (DBN). In order to achieve *effective* collaboration, the robot estimates the user's plan out of a set of *local* trajectories. During this process, the robot must propagate past driving information to the present in absence of global references. Therefore, we propose a solution to *local plan recognition* within the human-robot DBN through soft probabilistic constraints. These constraints model the uncertain information a potential user plan provides about the actually executed trajectory. We summarise such information in a *distance likelihood*, which we combine with other observations in order to derive the parameters of the human-robot DBN with the Expectation Maximisation algorithm. We illustrate the validity of the approach on a simulated wheelchair with a discrete interface as well as on a real wheelchair operated through a continuous joystick.

The second contribution is a set of probabilistic user models of driving. We introduce first an *analytical user model* of a user who cannot steer left. The *second user modelling approach* calibrates the user's driving style from real data through a general probabilistic machine learning technique called *Gaussian Process Regression*. We justify the selection of this technique with an analysis of user modelling requirements, and address how specific choices of the model fulfil them. In order to demonstrate the validity of the approach we analyse the driving behaviour of a user diagnosed with spastic-quadriplegia and recognise her navigation plans.

The third contribution is a framework for sharing the control of the wheelchair according to the estimated plans of its user. This framework is based on the Dynamic Window Approach. We propose three shared control algorithms where the user increasingly cedes more control to the robot. The first two shared control schemes correspond to standard navigation algorithms, namely *collision-avoidance* and *obstacle-avoidance*, which are special cases within our general framework. The third algorithm, Maximum A Posteriori (MAP) shared control, relies on the estimated plan of the user to navigate the wheelchair. We demonstrate the potential of user-adaptive shared control for a user who cannot steer left, but thanks to navigation assistance can reach goals at her left when she intends it. This experiment reveals that navigation assistance benefits from knowing the driving style of the wheelchair user, and that bi-directional communication in case of disagreement is necessary to *share the control* over the robot *effectively*.

Beknopte samenvatting

Uitdaging. De vergrijzing van de bevolking verandert overal ter wereld de demografie van de maatschappij. Als gevolg hiervan stijgt ook het aantal ouderdomskwalen, en dit terwijl er relatief gezien steeds minder verzorgers klaar staan om de ouderen en mindervaliden bij te staan. Deze evolutie is problematisch aangezien deze ouderdomskwalen vaak ook het verlies van mobiliteit teweegbrengen, waardoor deze bevolkingsgroep afhankelijk wordt van de hulp van anderen. Een beperkte mobiliteit kan ook leiden tot een geïsoleerd leven. De combinatie van beide effecten kan het welzijn en het zelfvertrouwen aantasten. Bovendien kan het verlies van mobiliteit ook een belangrijke economische impact hebben.

Feit. Technologie die de mobiliteit verbetert, zoals elektrische rolstoelen, brengt een oplossing voor deze demografische problematiek door mensen te helpen hun onafhankelijkheid te herwinnen. Elektrische rolstoelen zijn echter moeilijk te besturen en ongevallen met omgeving en met anderen mensen komen veelvuldig voor. Omwille van deze redenen wordt voor vele potentiële gebruikers geen rolstoel voorgeschreven indien ze de nodige cognitieve of fysieke vaardigheden missen. Zelfs bekwame gebruikers vinden het veilig besturen van een rolstoel moeilijk.

Opportuniteit. In de voorbije jaren kende de mobiele robotica een enorme vooruitgang. Autonome wagens legden grote afstanden af in de woestijn of navigeerden door een stad. Mobiele robotica vindt ook zijn weg naar alledaagse producten, zoals stofzuigers of grasmaaiers. Deze technologische vooruitgang laat veilige rolstoelnavigatie toe in bijna eender welke omgeving.

Uitdaging. De ouderen en mindervaliden vormen een zeer heterogene gebruikersgroep, met verschillende vaardigheden en noden. De uitdaging bestaat er dus in *robots op maat* te maken, die *doeltreffend samenwerken* met zulk een *heterogene gebruikersgroep*, met toch een *eenvoudige* en *intuïtieve* besturing.

Vereisten. Zulk een heterogene gebruikersgroep vereist een algemene methodologie die de navigatiebijstand kan aanpassen aan elk individu. Om frustratie bij de gebruiker te vermijden, moet de robot het navigatieplan van de gebruiker schatten op een manier die rekening houdt met de rijvaardigheden van de gebruiker. Bovendien mag de robot enkel bijstand verlenen wanneer de gebruiker dit nodig heeft. Gebruikers moeten immers gestimuleerd worden om zoveel mogelijk deel te nemen aan het navigatieproces om in goede mentale en fysieke conditie te blijven.

Methodologie. In deze thesis stellen we een *probabilistisch* raamwerk voor om *de navigatieplannen van de gebruiker te herkennen* op basis van een set van *lokale trajecten*. Om deze herkenning robuust te maken wordt ook de onzekerheid op de gebruikersplannen geschat. Het raamwerk brengt informatie uit het verleden samen met de specifieke rijstijl van de gebruiker om de waarschijnlijkheidsverdeling over gebruikersplannen te schatten. Bovendien worden deze geschatte plannen gebruikt om de controle over de rolstoel te delen. De robot garandeert de veiligheid van de gebruiker door te redeneren over botsingsvrije trajecten die ook fysisch uitvoerbaar zijn, rekening houdend met de rolstoelkinematica en -dynamica.

De eerste bijdrage van deze thesis is de modellering van mens-robot navigatie als een Dynamisch Bayesiaans Netwerk (DBN). Om *doeltreffende* samenwerking te realiseren, schat de robot het gebruikersplan uit een set van *lokale* trajecten. Tijdens dit proces moet de robot rij-informatie uit het verleden naar het heden overdragen, en dit in afwezigheid van globale referentiepunten. Daartoe stellen we een oplossing voor *lokale planherkenning* voor binnen het mens-robot DBN door het gebruik van zachte probabilistische beperkingen. Deze beperkingen modelleren de onzekere informatie die een mogelijk gebruikersplan verschaft over het eigenlijk uitgevoerd traject. We vatten deze informatie samen in een *afstandswaarschijnlijkheid*, die we combineren met andere waarnemingen om parameters van het mens-robot DBN te schatten met behulp van het Expectation-Maximisation algoritme. We illustreren de geldigheid van de aanpak zowel op een gesimuleerde rolstoel met een discrete interface als op een echte rolstoel met een continue joystick.

De tweede bijdrage bestaat uit een set van probabilistische gebruikersmodellen voor rolstoelbesturing. We bespreken eerst een *analytisch gebruikersmodel* voor een gebruiker die niet naar links kan sturen. De *tweede aanpak voor gebruikersmodellering* schat de rijstijl van een gebruiker op basis van echte data met behulp van een algemene probabilistische leertechniek die *Gaussiaans Proces Regressie* wordt genoemd. We rechtvaardigen het gebruik van deze techniek door aan te tonen hoe het aan de vereisten voor gebruikersmodellering voldoet. Om de geldigheid van de aanpak hard te maken analyseren we het rijgedrag van een gebruiker met spastische quadriplegie, en voorspellen we haar navigatieplannen.

De derde bijdrage is een raamwerk om de controle over de rolstoel te delen op basis van de geschatte gebruikersplannen. Dit raamwerk is gebaseerd op het Dynamisch Venster algoritme. We stellen drie gedeelde controle algoritmes voor. De eerste twee gedeelde controle algoritmes komen overeen met de standaard navigatiealgoritmes *botsingsvermijding* en *obstakelvermijding*, die elk op zich een speciaal geval zijn binnen ons algemeen raamwerk. Het derde algoritme, Maximum A Posteriori (MAP) gedeelde controle, maakt gebruik van het geschatte gebruikersplan om de rolstoel te besturen. We tonen het potentieel aan van gebruikersspecifieke gedeelde controle voor een gebruiker die niet naar links kan sturen, maar dankzij navigatiebijstand ook doelen gelegen aan de linkerkant kan bereiken. Dit experiment laat zien dat bidirectionele communicatie nodig is om *doeltreffend* de *controle over de robot te delen* in geval van onenigheid tussen robot en bestuurder.

Symbols, Definitions and Abbreviations

Abbreviations

ARD	: Automatic Relevance Determination
BN	: Bayesian Network
BCI	: Brain Computer Interface
DBN	: Dynamic Bayesian Network
DC	: Degree of Control
DWA	: Dynamic Window Approach
DW	: Dynamic Window
EM	: Expectation Maximisation
ESS	: Equivalent Sample Size
$\mathcal{GP}$	: Gaussian Process
GPS	: Global Positioning System
HMM	: Hidden Markov Model
MAP	: Maximum A Posteriori
ML	: Maximum Likelihood
MOVEMENT	: *MO*dular *VE*rsatile *M*obility *EN*hancement *T*echnology
NV	: Network Variable (wheelchair hardware control units)
pdf	: Probability density function
cdf	: cumulative distribution function
RADHAR	: *R*obotic *AD*aptation to *H*umans *A*dapting to *R*obots
RT	: Reaction Time
RQ	: Rational Quadratic
RV	: Random Variable
SE	: Squared Exponential
SLAM	: Simultaneous Localisation and Mapping

Medical terminology

AD	: Alzheimer Disease
ALS	: Amyotrophic Lateral Sclerosis
EMG	: Electromyography
EDA	: Electro Dermal Activity
CP	: Cerebral Palsy
CVA	: Cerebrovascular Accident
MS	: Multiple Sclerosis
MSA	: Multiple System Atrophy

Generic symbols

Scalars, vectors, matrices

a	: scalar
$\boldsymbol{a}$	: vector
$\|\boldsymbol{a}\|$	: Euclidean norm of a vector
$\boldsymbol{A}$	: matrix or collection of vectors
$\boldsymbol{a}_{1:k}$	: Time series in $\boldsymbol{a}$, i.e. $\boldsymbol{a}_{1:k} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k]$

Sets

$\mathcal{X}$	: generic set
$\sharp(\mathcal{X})$	: cardinality of set $\mathcal{X}$
$\mathcal{A} \setminus \mathcal{B}$	: set difference
$\mathcal{A} \cap \mathcal{B}$	: set intersection
$\mathcal{A} \cup \mathcal{B}$	: set union
$\varnothing$	: empty set

Miscellaneous symbols

$j\omega$	: frequency in radians
$\Pi([x_1, x_2])$	: Heaviside step function over the interval $[x_1, x_2]$
$\delta(\boldsymbol{x} - \boldsymbol{x}_0)$	: Dirac's delta function centred at $\boldsymbol{x}_0$

Probabilistic models

Symbols

$p(\boldsymbol{a})$	: Probability of $\boldsymbol{a}$ or pdf of $\boldsymbol{a}$, depending on the context
$\boldsymbol{a} \sim p(\boldsymbol{a})$	: RV $\boldsymbol{a}$ is distributed according to $p(\boldsymbol{a})$
$p(\boldsymbol{a}) \propto p(\boldsymbol{b})$	: probability $p(\boldsymbol{a})$ is proportional to probability $p(\boldsymbol{b})$
$\mathbb{E}[\boldsymbol{a}]$	: expected value of $\boldsymbol{a}$
$\boldsymbol{a} \perp\!\!\!\perp \boldsymbol{b}$	: RVs $\boldsymbol{a}$ and $\boldsymbol{b}$ are marginally independent
$\boldsymbol{a} \perp\!\!\!\perp \boldsymbol{b} \mid \boldsymbol{c}$	: RVs $\boldsymbol{a}$ and $\boldsymbol{b}$ are conditionally independent given $\boldsymbol{c}$.
$p(\boldsymbol{a} \mid \boldsymbol{b})$	: conditional probability of $\boldsymbol{a}$ given $\boldsymbol{b}$ or conditional pdf of $\boldsymbol{a}$ given $\boldsymbol{b}$, depending on the context

Probability distributions

$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	: $\boldsymbol{x}$ has a Gaussian distribution of mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathbb{G}(\boldsymbol{x})$	: cumulative distribution function (cdf) of a standard normal RV of mean 0 and standard deviation 1, i.e. $\mathcal{N}(x \mid \mu = 0, \sigma = 1)$
$\mathit{Dirac}(\boldsymbol{x} \mid \boldsymbol{x}_0))$	: $\boldsymbol{x}$ has a Dirac distribution centred at $\boldsymbol{x}_0$. The Dirac distribution assigns a point-probability mass to a continuous space, $\mathit{Dirac}(\boldsymbol{x} \mid \boldsymbol{x}_0) = \delta(\boldsymbol{x} - \boldsymbol{x}_0)$
$\mathit{Rayleigh}(\boldsymbol{x} \mid \sigma)$	: $\boldsymbol{x}$ has a Rayleigh distribution of parameter σ
$\mathit{Mult}\big(\boldsymbol{m} \mid N, \boldsymbol{p}\big)$	: the counts $\boldsymbol{m}$ have a multinomial distribution. Each count represents the number of occurrences of a discrete RV with K states, where the states have probabilities $\boldsymbol{p} = [p_1 \dots p_K]$ and N is the total number of trials.
$\mathit{Dirichlet}\big(\boldsymbol{p} \mid \boldsymbol{\alpha}\big)$	: the K-dimensional probability vector $\boldsymbol{p}$ has a Dirichlet distribution of parameter $\boldsymbol{\alpha}$

Bayesian Networks and Dynamic Bayesian Networks

$\mathcal{U}$	: universe of all variables in a Bayesian Network
$\boldsymbol{e}$	: hard evidence vector
$\leftarrow A \rightarrow$	: diverging or tail-to-tail connection in a BN or DBN
$\rightarrow A \rightarrow$	: serial or head-to-tail connection in a BN or DBN
$\leftarrow A \leftarrow$	: serial or head-to-tail connection in a BN or DBN
$\rightarrow A \leftarrow$	: converging or head-to-head connection in a BN or DBN
ϕ_i	: potential over a discrete set of variables
$\mathit{domain}(\phi_i)$	: variables over which the potential is defined
$\boldsymbol{\Phi}_X$	: potentials with X in their domain, i.e. $\{\phi_i \in \boldsymbol{\Phi} \mid X \in \mathit{domain}(\phi_i)\}$

$(\prod \mathbf{\Phi})^{\downarrow A_i}$: marginalisation of a set of potentials $\mathbf{\Phi}$ down to variable A_i, i.e. to eliminate all variables except A_i. Mathematically $(\prod \mathbf{\Phi})^{\downarrow A_i} = \sum_{\mathcal{U} \setminus A_i} \prod \mathbf{\Phi}$

ψ^S : outgoing message

$(BN_1, BN_{\rightarrow})$: representation of a DBN. BN_1 is a BN of the first time slice. $BN_{\rightarrow}$ is a two-slice temporal BN.

Gaussian Process Regression

$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x\prime})\right)$:
Function values $f(\boldsymbol{x})$ are distributed according to the Gaussian Process $\mathcal{GP}$ of mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x\prime})$

$m(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right]$: mean function of a $\mathcal{GP}$

$k(\boldsymbol{x}, \boldsymbol{x\prime}) = \mathbb{E}\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f(\boldsymbol{x\prime}) - m(\boldsymbol{x\prime})\right)\right]$:
positive definite covariance function of a $\mathcal{GP}$

Probabilistic human-robot navigation

Variables

$\boldsymbol{a}$: Robot action, which can be continuous or discrete. In this work, robot actions move the robot physically

$\boldsymbol{i}$: User's navigation plan or user's navigation intention

$\boldsymbol{\mathcal{I}}_k$: User plan library, which denotes all collision-free user plans $\boldsymbol{i}_k$ at time k

$\boldsymbol{m}$: Estimated environment map

$\boldsymbol{M}$: All environment information, i.e. prior maps, estimated maps and external sensor readings

$\boldsymbol{t}_{goal}$: goal twist, i.e. $\boldsymbol{t}_{goal} = [v_{goal}\ \omega_{goal}]^T$

$\boldsymbol{u}$: User inputs, which can be continuous or discrete

v : linear velocity

ω : angular velocity

$\boldsymbol{x}$: Robot pose, including position (x, y) and orientation θ

$\boldsymbol{z}$: Sensor signals to perceive the external environment or the robot's own motion

Metrics

$d_{Hausdorff}(\mathcal{P}, \mathcal{Q})$: Hausdorff distance between curves $\mathcal{P}$ and $\mathcal{Q}$
$\vec{\delta}(\mathcal{P}, \mathcal{Q})$: directed Hausdorff distance between curves $\mathcal{P}$ and $\mathcal{Q}$
$d_{Fréchet}(\mathcal{P}, \mathcal{Q})$: Fréchet distance between curves $\mathcal{P}$ and $\mathcal{Q}$
$d_{MD}(\boldsymbol{x}, \boldsymbol{y})$: Martinez-Duffy metric between poses $\boldsymbol{x}$ and $\boldsymbol{y}$
$SE(3)$: Special Euclidean group(3)

Probabilistic models of plan recognition

$\mathcal{H}_{1:k}$: prior history of user signals ($\boldsymbol{u}$), robot actions ($\boldsymbol{a}$), and constant parameters ($\boldsymbol{h}$), i.e. $\mathcal{H}_{1:k} = \{\boldsymbol{u}_{1:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{h}\}$

$bel(\boldsymbol{i}_k) \equiv p_{post}\big(\boldsymbol{i}_k \mid \boldsymbol{u}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big)$:
posterior belief in user plan $\boldsymbol{i}_k$ after integrating all available knowledge up to time k

$p_{prior}(\boldsymbol{i}_k)$: prior belief in user plan $\boldsymbol{i}_k$ before considering the user's current input $\boldsymbol{u}_k$

$p_{gen}\big(\boldsymbol{x}_j^i \mid \boldsymbol{x}_{j-1}^i, \ldots, \boldsymbol{x}_1^i\big)$:
Generative model to assign probability mass to discrete poses of user plan $\boldsymbol{i}$

$p\big(\boldsymbol{x}_k\big)$: localisation uncertainty

$p_{dist}\big(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k\big)$: distance likelihood, which models the likelihood of user plans $\boldsymbol{\mathcal{I}}_k$, given their distances to the robot's future trajectory $\boldsymbol{x}_{k:end}$

$p_{\exp}(d \mid \boldsymbol{i}_k)$: Exponential distance likelihood

$p_{\arctan}(d \mid \boldsymbol{i}_k)$: Arc-tan distance likelihood

$p_{\cos}(d \mid \boldsymbol{i}_k)$: Cosine distance likelihood

$p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big)$:
plan process function, which is a probabilistic model of the temporal transition between user plans in response to robot actions.

$p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k)$: user model, which represents the likelihood of current user input signals $\boldsymbol{u}_k$, given the user has plan $\boldsymbol{i}_k$ in mind

$p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_{k-1:k})$: Extended user model, which includes a finite memory of prior user plans.

$p_{perf}\left(\boldsymbol{u}_k^{dis} | \boldsymbol{u}_k^{ref}\right)$: performance model, which relates the distorted user input $\boldsymbol{u}_k^{dis}$ in presence of steering disabilities to the reference input $\boldsymbol{u}_k^{ref}$

Shared control

$\boldsymbol{dwa}(k)$: dynamic window at time k, i.e. dynamically feasible twists $\boldsymbol{v}$ within the next control interval

$\boldsymbol{adm}(k)$: admissible twists at time k, i.e. twist which allow the robot to stop in time

Δv_{DW}^{min} : minimal extension of the DW in the linear velocity

$\Delta\omega_{DW}^{min}$: minimal extension of the DW in the angular velocity

$cost(\boldsymbol{v})$: total cost function of a twist $\boldsymbol{v}$

$c_{user}(\boldsymbol{v})$: user cost of a twist $\boldsymbol{v}$

$c_{path}(\boldsymbol{v})$: path-following cost of a twist $\boldsymbol{v}$

$c_{coll}(\boldsymbol{v})$: cost of a twist $\boldsymbol{v}$ because of collisions at high speed

$c_{ca}(\boldsymbol{v})$: collision-avoidance cost of a twist $\boldsymbol{v}$

$c_{oa}(\boldsymbol{v})$: obstacle-avoidance cost of a twist $\boldsymbol{v}$

α_{user} : weight of the user cost

α_{path} : weight of the path-following cost

α_{coll} : weight of the cost because of collisions at high speed

α_{ca} : weight of the collision-avoidance cost

α_{oa} : weight of the obstacle-avoidance cost

$\Delta t_i^{travel}(\boldsymbol{v})$: travelling time of twist $\boldsymbol{v}$ on the closest collision-free trajectory. The definition of proximity depends on the type of shared control

$\boldsymbol{i}_k^{MAP}$: MAP estimate of the user's plan

$c_{grad}(\mathbf{v})$: cost of twist $\mathbf{v}$ for tracking the MAP plan $\boldsymbol{i}_k^{MAP}$

α_{grad} : weight of the cost for tracking the MAP plan $\boldsymbol{i}_k^{MAP}$

$H\left(\omega^{dis} \mid \omega^p = \hat{\omega}^p\right)$:
conditioned entropy of the angular component of the analytical user model. The entropy is conditioned on the predicted angular velocity $\hat{\omega}^p$ for a certain user plan.

Contents

Preface iii

Abstract vii

Beknopte samenvatting ix

Symbols, Definitions and Abbreviations xi

Contents xvii

1 Introduction 1

1.1 An architecture for driver navigation assistance 2

1.2 Generalization to other applications 6

1.3 Overview and contributions 7

1.3.1 Probabilistic models and tools 7

1.3.2 Plan recognition 7

1.3.3 User modelling 8

1.3.4 Shared control 8

2 Probabilistic Models and Tools 11

2.1 Bayesian Networks 11

2.1.1 Static Bayesian Networks 11

2.1.2 Inserting evidence 13
2.1.3 Exact Inference: The Junction-Tree Algorithm 14
2.1.4 Dynamic Bayesian Networks 19
2.2 Gaussian Processes 21
2.2.1 Model description 21
2.2.2 Bayesian model selection 23
2.2.3 Sparse Gaussian Processes 27

3 Plan Recognition **29**
3.1 General description of plan recognition 30
3.1.1 Plan recognition methodology 31
3.1.2 Restricting hypotheses 31
3.1.3 User adaptation 32
3.1.4 Bayesian approaches 33
3.1.5 Robustness 35
3.1.6 Applications 35
3.2 Probabilistic Human-Robot Navigation 36
3.2.1 Simultaneous Localisation and Mapping (SLAM) 39
3.2.2 Plan recognition and user modelling 42
3.2.3 Generation of user plans 51
3.2.4 Local plan recognition 54
3.2.5 Summary of the models for plan-recognition 66
3.2.6 History of the framework for plan recognition 70
3.2.7 Related work 70
3.3 Experimental results 73
3.3.1 Plan recognition for a discrete user interface 73
3.3.2 Continuous interface 89
3.4 Conclusions 91

4 User models for wheelchair driving 93

4.1 Potential user group of robotic wheelchairs 94

4.1.1 Pathologies and injuries of the nervous system 95

4.1.2 Symptoms affecting wheeled mobility 100

4.2 Manual control 103

4.2.1 Quasi-linear model of the human operator 106

4.2.2 Optimal control model 109

4.2.3 Adaptation to changing task requirements 111

4.2.4 Reaction times 113

4.3 User models for wheelchair driving 116

4.3.1 First-principles linear regression user model 119

4.3.2 Analytical user model for users with weak directional steering capabilities 121

4.3.3 Gaussian Process user model with local trajectories 127

4.3.4 Experiments: user modelling and plan recognition for a user diagnosed with spastic quadriplegia 145

4.4 Conclusions 198

5 Shared wheelchair control 201

5.1 Shared Control Algorithm 202

5.1.1 Representation of the wheelchair's dynamics 202

5.1.2 Dynamic Window Approach 204

5.1.3 Related work 212

5.2 Collision- and obstacle-avoidance respecting kinematic and dynamic constraints 223

5.2.1 Collision avoidance 223

5.2.2 Obstacle avoidance 232

5.3 Maximum-A-Posteriori (MAP) shared control 238

5.3.1 Safe navigation based on the user's plan 238

5.3.2 Adaptive decision making 240

5.3.3 Experiments 242
5.3.4 Discussion 245
5.4 Conclusions 246

6 Conclusions **249**
6.1 Contributions 250
6.1.1 Plan recognition 250
6.1.2 User modelling 252
6.1.3 Shared control 253
6.1.4 User's perspective on navigation assistance 254
6.2 Future work 256
6.2.1 Plan recognition 256
6.2.2 User modelling 258
6.2.3 Shared control 259

References **261**

Curriculum vitae **271**

List of publications **273**

Chapter 1

Introduction

"Zooropa... Don't worry baby, it's gonna be alright
Zooropa... Uncertainty... can be a guiding light"

Bono, The Edge, Adam Clayton, Larry Mullen Jr. *U2*
album: *Zooropa*. song: Zooropa (1993)

Automated assistive devices like robot arms, robotic guide dogs for the blind, robotic walkers and wheelchairs are designed to make the life of elderly and disabled people more comfortable by helping them with every-day-life activities. These assistive devices are meant to increase the independence of people, to enhance their quality of life, and to provide more comfort and safety. One of the crucial requirements for such an independent life is an enhanced mobility. Many people obtain this mobility through electric wheelchairs.

Unfortunately, not all people are able to control a wheelchair properly. A recent report (Fehr et al., 2000) suggests that, in some cases, electric wheelchairs are not prescribed due to the patient's lack in motor or cognitive skills. Even seemingly easy manoeuvres such as driving through a door, following a corridor, or avoiding obstacles are experienced as difficult, time-consuming and (therefore) annoying. The relatively large size of wheelchairs compared to the dimensions of places in people's homes or in hospitals, makes navigation even more difficult. As a consequence, Fehr's survey indicates the existence of a market for robotised wheelchairs, which help in difficult manoeuvres.

In accordance to current society trends and the claim for robotised devices, many "smart" wheelchairs have been developed in the last decades, for instance: SmartChair (Parikh et al., 2004), Wheelesley (Yanco, 1998), Senario (Katevas et al., 1997), Navchair (Simpson et al., 1998; Simpson and Levine, 1999), and

Sharioto (Demeester, 2007; Vanhooydonck, 2007), just to name a few (see also the overview in (Special Issue RAM, 2001)). Often, the developed architectures consist of different algorithms that each realize a specific assistance behaviour, such as *drive-through-door*, *follow-corridor*, or *avoid-collision*. In most cases, different levels of shared control are defined, ranging from full operator control to full computer control.

The approach to *human-robot navigation* followed here differs from previous automated wheelchair approaches in four major aspects.

Firstly, the user's driving behaviour is modelled in order to estimate her plans in real-time, which allows to adapt to different user driving behaviours, different interface types and different wheelchair types.

Secondly, the basic navigation assistance algorithm consists of a fine motion planner, which allows to execute even those manoeuvres for which heuristic algorithms usually fail.

Thirdly, the user's intentions or plans are estimated explicitly as well as the uncertainty on these estimates. For this, a probabilistic approach is adopted that combines past plan estimates and environment information with observations of the user's driving style obtained through her inputs. Considering the inherent uncertainty is invaluable when taking decisions to aid the user. As the robot estimates the user's plan (also called *plan recognition*, *intent recognition*, or *intention estimation*), the user does not have to specify which navigation task should be executed. Instead, the robot estimates this navigation task itself and decides which and how much assistance is necessary for each situation. Therefore, we claim that plan recognition enables the user to control the wheelchair in a more intuitive way by using the interface she is used to.

Fourthly, user plans are represented as trajectories from the current robot state to a certain goal state instead of modelling user plans as separate tasks such as *follow-corridor* or *drive-through-door*. This formulation is more general and allows assistance for *any* conceivable manoeuvre.

1.1 An architecture for driver navigation assistance

This section introduces our probabilistic paradigm for *human-robot navigation*. It describes the collaboration between human and robot when performing together a navigation task. The user will perceive the robot as intuitive if she has the impression that the robot understands her intentions and only provides the necessary amount of driving assistance. Only then will she be prepared to share part of her autonomy with the robot in exchange for safer driving. Therefore, the robot needs to estimate the user's navigation plans, which will be modelled as trajectories that the robot should execute. Based on the estimated trajectory, a robust decision can be made for navigation assistance. The robot's behaviour can be made inherently

safe by taking only safe trajectories into account as possible driver intentions, i.e. trajectories that are collision-free and that account for the robot's kinematics and dynamics.

Figure 1.1 shows the architecture we propose for navigation assistance. Its essential property is the existence of multiple control loops of which the human operator forms an integral part. When a human interacts with a robot she intends to achieve a certain goal and adapts her strategy to the observed behaviour of the robot. We will analyse the architecture from two perspectives, a human-centric and a robot-centric view.

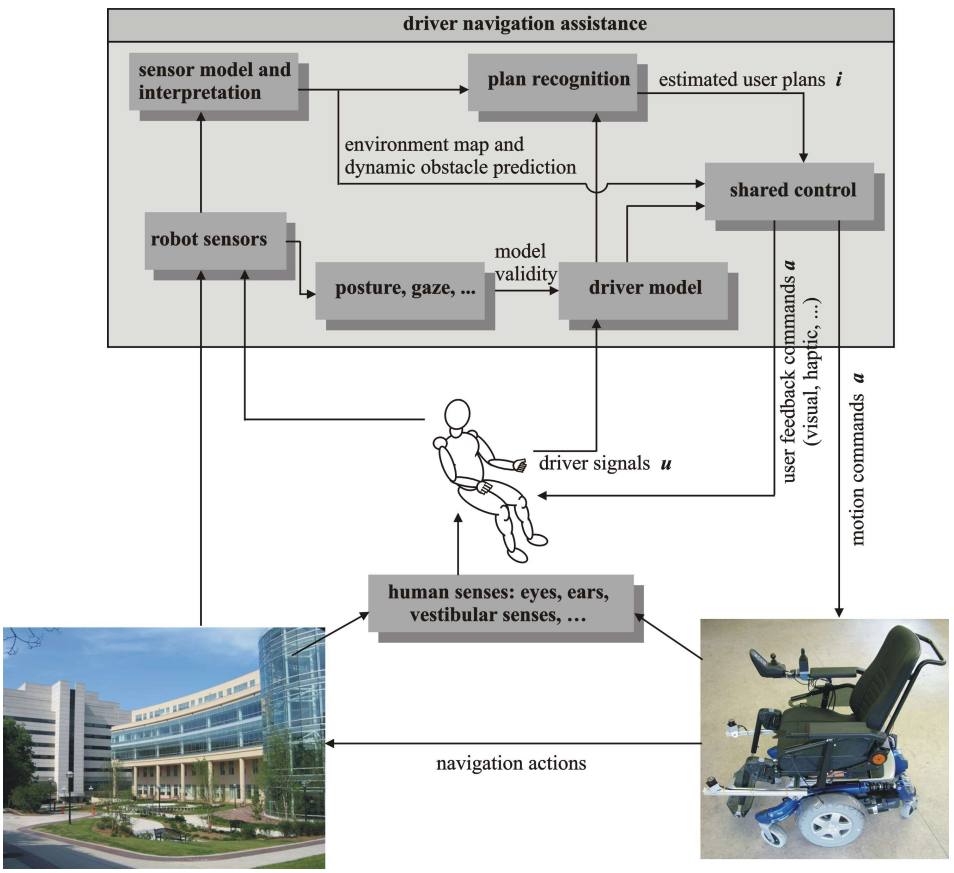


FIGURE 1.1: Framework for driving assistance

Human view of navigation. At the centre of the human-robot navigation paradigm stands the user. The human perceives the environment through her senses (vision, hearing and vestibular systems) before deciding what to do. The user

elaborates then a plan to reach a certain destination, but she cannot communicate her plan explicitly to the robot. Instead, she must transform her intentions into a series of user inputs $\boldsymbol{u}$ through an adapted interface.

Assistive technology offers a myriad of interfaces for controlling power wheelchairs because these interfaces need to match the user's abilities. Typical examples include: two dimensional joysticks (Curtis, 2010; Dynamic, 2010), switch-based discrete inputs (Permobil, 2010), head-operated and chin-operated joysticks (Permobil, 2010), sip-and-puff devices (Therafin, 2010), infra-red position sensors (HMC International, 2010), or tongue switches. Despite the multitude of available interfaces, controlling an electric wheelchair is difficult and potentially unsafe.

(Arshak et al., 2006) offers an overview of candidate wheelchair interfaces depending on the level of spinal cord lesion. Depending on the type of impairment, the only possible interface might have very low information capacity. The user might face then considerable difficulty in steering a wheelchair safely or her freedom of movement might be severely constrained. Controlling a wheelchair with a joystick, which is the richest control interface, can prove difficult as well due to the wheelchair's large size compared to the available space in indoor environments. The benefit of human-robot navigation is that it can complement the driving abilities of a user, and thus enlarge the potential group of power wheelchair users.

In the architecture of figure 1.1 the specific interface is abstracted away from the rest of the system by the *driver model*, which represents how a particular user translates her plans into robot inputs by means of a specific interface. This model will be discussed later as part of the robot control loop.

User inputs $\boldsymbol{u}$ are translated by the robot into motion commands $\boldsymbol{a}$, which causes robot to move in the environment. The human operator perceives this motion through her senses and reacts by eventually adapting her navigation strategy. This closes the main human-robot control loop, which includes the user as central element.

Robotic view of navigation. After the human-centric description, we change perspective and look at the problem from the side of the robot. We will describe the block labelled *driver navigation assistance* in figure 1.1. The robot is cooperating with its human operator during navigation. For the robot its operator is another element of the environment it needs to sense and understand. Therefore, one part of its sensors point to the external environment and other sensors serve to perceive the user. We denote *external sensor* inputs as $\boldsymbol{z}$. They might include stereo vision, laser scanners, time-of-flight-cameras, ultrasound, radar, odometry sensors, inertial measurement units, and GPS. In the preceding list we include both sensors to perceive external objects as well as sensors to measure the robot's ego-motion. Since no single sensor modality can perceive unknown environments under all conditions, the most robust approach is to combine different sensor types based on different physical principles.

The second class of sensors are *human-oriented*. They might also include cameras, but focusing on the user instead of viewing the environment. EMG sensors, accelerometers placed on the user, or sensors to measure stress level by electrodermal activity (EDA) (Setz et al., 2010) can allow the robot to asses the state of its user. The most relevant *human-oriented* sensor, however, is the user interface by which the user communicates her plans to the robot.

On top of the sensor layer in the architecture of figure 1.1, the robot builds its cognitive capabilities. On one hand, the robot needs to interpret the incoming stream of sensor data in order to construct a map of a potentially unknown environment, and to localise within this map. In robotics this problem is called *Simultaneous Localisation and Mapping (SLAM)*. Localisation and map-building capabilities are invaluable to reason about the user's plan as we will explain next. Environment perception is thus an input to the *plan recognition* component in the navigation assistance architecture.

Another interesting source of information for the robot is the estimation of dynamic obstacles, which should be avoided. For instance, (Bennewitz et al., 2005; Bennewitz et al., 2002), propose an approach to detect and predict motion patterns of people based on sensory input of cameras and laser-range finders. Such information has a direct impact on the navigation decisions of the robot. Therefore, in figure 1.1 *the sensor model and interpretation* block is linked to the *shared control* component.

Robot performance cannot be measured in isolation from its operator because the human is effectively closing the robot's control loop. Instead, human and robot must share a common understanding of each other's capabilities. Further, intuitive human-robot collaboration is only possible if robot and operator are also aware of each other's plans. Otherwise, the user might become frustrated instead of being helped if the robot consistently disregards her plans. In this sense, the probabilistic *plan recognition* component in figure 1.1 must recognize all possible user navigation intents, estimate the uncertainty on these intents, and make the estimation adaptive to the specific user that is interacting with the robot. In order to realize this, a representation of user intents is chosen that is different from previous approaches, which nearly all represent intents in terms of specific assistance algorithms such as *follow-corridor* or *avoid-obstacle*. The general *plan recognition* methodology is the topic of chapter 3.

Plan recognition relies on specific *driver models* to adapt the estimation of the user's plan to her specific abilities. This model captures how a user transforms her mental plans into interface inputs and will be discussed at length in chapter 4. The experimental section in chapter 4 reports user modelling and plan recognition results for a user diagnosed with spastic-quadriplegia.

The last component of the *driver navigation assistance* is the *shared control* block. Shared control is the link between perception and action. It combines information of the environment, such as the trajectories of dynamic obstacles, with the perception of the user in order to compute optimal robot actions. Robot actions can directly actuate the user's interface as haptic feedback, or can correspond to navigation

commands that move the robot. Haptic feedback opens a faster communication channel to the user, who no longer needs to wait for the movement of the robot in order to estimate the robot's intentions. Motion commands, on the other hand, should ensure safe and comfortable navigation, while corresponding to the user's plans. Chapter 5 presents our implementation of a *shared control* framework.

1.2 Generalization to other applications

Although, the subject of this text is wheelchair navigation, the framework generalises to other human-robot collaboration scenarios as well. Changes are only necessary in very specific parts of the architecture, such as in the representation of user plans, in the interfaces between human and robot, or in the set of available robot actions.

Human-robot navigation can be interesting for other transportation means. For instance, drivers of cars, trucks, and bulldozers might equally benefit from navigation assistance that ensures their safety. Those vehicles and the wheelchair share the limited acceleration and deceleration capabilities, which are explicitly considered in the framework. Therefore, the navigation paradigm can be extended to other navigation means with limited accelerations. Also, the wheelchair's non-holonomic character and relatively large size, makes parking manoeuvres for wheelchairs as challenging as for cars. The use of a joystick versus a steering wheel and gas pedal is considered to be a difference in user interface, which is abstracted away from the rest of the framework through the user model. Furthermore, in construction equipment for example, the use of joysticks is becoming more and more common place.

Plan recognition and *driver navigation assistance* is even expected to be more challenging in the case of wheelchair navigation. For wheelchair driving, the environment is more unstructured. For example, there is a lack of traffic signs and rules. This makes it more difficult to estimate the navigation intention of drivers.

Furthermore, the wheelchair driver population is very heterogeneous. Their cognitive or physical capabilities can be limited, for instance, by poor traffic awareness, visual inattention, neglect, poor concentration, tremors, spasms, and limitations in range of motion or power. Moreover, the driver's behaviour can change over time. In the short term these changes originate from stress or fatigue, and in the long term the driver might learn to driver better or her condition might deteriorate. Therefore, elderly and disabled drivers constitute an extremely challenging user group.

Wheelchair manoeuvring is complex. In contrast to car navigation, wheelchair navigation often takes place in unconstrained and complex environments, populated with dynamic objects that must be avoided. Manoeuvres are to be undertaken frequently. Often the dimensions of the wheelchair leave only very limited manoeuvring space. Disturbances from uneven terrain, accelerations or

decelerations can interfere with the user's commands by upsetting the navigational quality and possibly lead to instability.

For this reason, if the probabilistic framework for human-robot navigation allows for efficient and safe co-operation with a challenging user population in a complex domain such as wheelchair navigation, it is believed that this framework can generalise to other domains and user groups as well.

1.3 Overview and contributions

This section provides an overview of the thesis and summarises the main contributions along the three main components of the probabilistic framework for *human-robot navigation*: plan recognition, user modelling and shared control.

1.3.1 Probabilistic models and tools

Chapter 2 provides a summary of the probabilistic tools employed in the remainder of the thesis.

1.3.2 Plan recognition

Chapter 3 reviews first the problem of plan recognition in other domains in order to set the context. The following section defines human-robot navigation in probabilistic terms. A robot interacting with a human is moving through potentially unknown environments, in which it needs to localise. Further, the human operator does not communicate her intentions explicitly. She interacts with the robot through her conventional interface, which might provide only a limited amount of information about her true plans. Therefore, the robot needs to recognise the plan of its user in addition to localising in an unknown map.

The robot's state estimation problem is divided into two sub-tasks, Simultaneous Localisation and Mapping (SLAM) and plan recognition. After a brief review of the SLAM problem, section 3.2.2 describes a probabilistic framework for plan recognition. This framework proposes a representation of user plans, which differs from previous approaches that nearly all represent intentions in terms of specific assistance algorithms. Both global and local plans are discussed in term of their implications on the plan recognition process. The chapter concludes with an experimental section, which describes plan recognition results for a discrete and a continuous robotic interface.

The main contribution of chapter 3 is a procedure to calibrate the parameters of the plan recognition framework for a set of local user plans. These plans are beneficial for immediate navigation assistance at the cost of increasing the complexity of

plan recognition. They lack any reference to global destinations, which would ease the propagation of probability in time. Therefore, an alternative is needed to reason about the temporal evolution of local user plans. The solution to local plan recognition is formulated as an optimisation with soft constraints. These constraints originate from the resemblance between each user plan and the future path of the robot, which provides a kind of global reference. Resemblance is measured in terms of appropriate trajectory metrics and the soft constraints are enforced through a probabilistic observation likelihood.

1.3.3 User modelling

Chapter 4 offers three perspectives of a human operating a wheelchair. First, we segment the potential user population, to understand the needs of the users who could benefit from navigation assistance. The second view is system-theoretic as wheelchair driving corresponds to a manual control situation. However, this view is insufficient to characterise a wheelchair user completely. Besides compensating perceived error signals, wheelchair driving also involves high-level cognitive and preview processes. The divide between traditional manual control and high-level cognition is bridged in a third, probabilistic, perspective. The probabilistic view captures both modelling uncertainty and inherent stochastic noise.

The goal of the probabilistic view is to calculate the user signals' likelihood in function of a user plan library. This likelihood, called *user model*, allows to relate the user's current input to a set of hypotheses in a way that sheds light on the user's navigation plans.

The main contributions of the chapter are two user modelling approaches. The first approach is an analytical model of driving disabilities, which is applied to a user who is incapable of steering left. This user model is mainly of theoretical interest in order to demonstrate the benefits of navigation assistance in chapter 5.

The second user modelling approach is based on a non-parametric probabilistic machine learning technique. This technique can model driving patterns of arbitrary users and can enhance the understanding of the process of controlling a wheelchair. The power of this general user model is illustrated on a case study, which analyses the driving performance of a real user diagnosed with spastic-tetraplegia.

1.3.4 Shared control

Intuitive human-robot navigation is only possible if the robot understands its user and respects her wishes. Since in most cases the user does not communicate her plans explicitly, the robot needs to estimate them. Estimated plans have to be translated then into robot actions. This implies a choice between different possible user plans. Chapter 5 proposes a shared control framework to formalise the notion

of choice between safe robot actions that respect the wheelchair's kinematics and dynamics.

Chapter 5 describes how different human-centric desiderata, such as *collision-avoidance*, *obstacle-avoidance* or tracking the estimated plan of a user can be formalised as dynamic window approaches. In all cases, the basic building block for the different shared control algorithms are a set of local or global paths employed to recognise the intent of the user. The shared control framework is sufficiently flexible to accommodate any plan library. The only requirement is that these plans are safe collision-free paths that respect the robot's kinematics. The dynamic window is the means to incorporate the robot's dynamics in navigation assistance.

The benefit of user-adapted shared control is demonstrated on a case study with a user who cannot steer to the left. Through navigation assistance, this user is able to reach locations at her left when she intends to do so.

Chapter 2

Probabilistic Models and Tools

> *"You see, one thing is, I can live with doubt and uncertainty and not knowing. I think it's much more interesting to live not knowing than to have answers which might be wrong. I have approximate answers and possible beliefs and different degrees of certainty about different things, but I'm not absolutely sure of anything and of many things I don't know anything about, such as whether it means anything to ask why we're here, and what the question might mean. I might think about it a little bit and if I can't figure it out, then I go on to something else, but I don't have to know an answer, I don't feel frightened by not knowing things, by being lost in a mysterious Universe without having any purpose, which is the way it really is so far as I can tell, possibly. It doesn't frighten me."*
>
> Richard P. Feynman ($\star$1918 – †1988)
> BBC interview in the program *Horizon*, 1981

In the remainder of the thesis we propose a robot that is not afraid of "doubt and uncertainty", but prefers to measure its lack of knowledge. Therefore, the goal of this chapter is to introduce a collection of probabilistic models, which allow to quantify uncertainty during human-robot navigation.

2.1 Bayesian Networks

2.1.1 Static Bayesian Networks

A Bayesian Network (Pearl, 1988) is a directed acyclic graph over a set of variables, $\mathcal{U} = \{A_1, \ldots, A_n\}$. These variables are connected by directed edges without loops, i.e. there is no path following the direction of the arrows with the same node as

start- and endpoint. Further, each variable A_i takes values from a finite set of mutually exclusive states, $\{a_i^1, \ldots, a_i^{m(A_i)}\}$.

Figure 2.1 shows the graphical representation of a Bayesian Network over variables $\mathcal{U} = \{A_1, \ldots, A_9\}$. Some nodes in figure 2.1 are shaded in grey, whereas others are uncoloured. In the graphical representation, shaded nodes are observed, i.e. their values are known. Hidden nodes, on the contrary, appear uncoloured. We will discuss in the following sections how to query a Bayesian Network for the probability of such hidden nodes given different types of evidence.

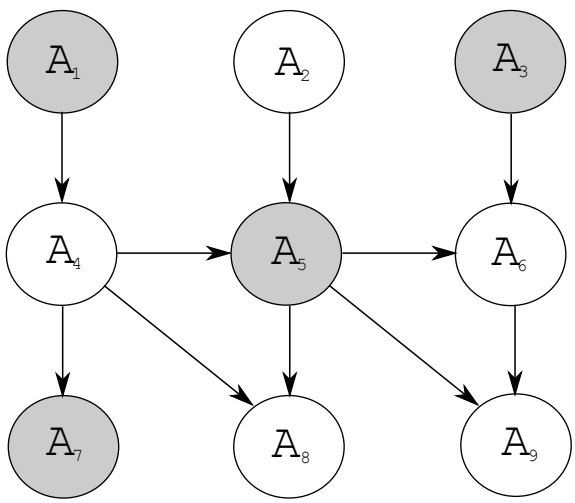


FIGURE 2.1: Example of a Bayesian Network over variables $\mathcal{U} = \{A_1, \ldots, A_9\}$. Shaded nodes are observed and the rest represent hidden nodes, whose values are unknown.

A Bayesian Network defines a factorisation of the joint probability over the set of all variables over which it is specified.

$$p(\mathcal{U}) = \prod p(A_i|parents(A_i)) = p(A_1|parents(A_1)) \cdots p(A_n|parents(A_n)) \quad (2.1)$$

where $parents(A_i)$ represents the set of nodes in the Bayesian Network whose arrows point in the direction of A_i and immediately precede node A_i. The conditional probability table $p(A_i|parents(A_i))$ is a function $\phi : parents(A_i) \times A_i \rightarrow [0, 1]$ over the domain $domain(\phi) = \{A_i, parents(A_i)\}$.

The factorisation of equation 2.1 implies a series of independence assumptions for a particular Bayesian Network. These independence relationships are also reflected in the graphical representation of the Bayesian Network. Two sets of variables, X and Y, are conditionally independent given a third set Z, if every path between X and Y is blocked by Z. Then, variable sets X and Y are d-separated (Pearl, 1986). Changes in certainty of d-separated variables have no influence on each other. It

remains to specify how a set of variables Z can block paths from X to Y. In a Bayesian Network, directed paths can reach a variable A in three configurations:

1. diverging connection or tail-to-tail: $\leftarrow A \rightarrow$
2. serial connection or head-to-tail: $\rightarrow A \rightarrow$ or $\leftarrow A \leftarrow$
3. converging connection or head-to-head: $\rightarrow A \leftarrow$

Diverging and serial connections at variable A block a path between X and Y if variable A is observed, i.e. if it belongs to set Z. In figure 2.1 the path $A_2 \rightarrow A_5 \rightarrow A_8$ is blocked by the serial connection at A_5 and the path $A_8 \leftarrow A_5 \rightarrow A_9$ is blocked by the diverging connection at A_5. Converging connections, on the other hand, block a path if neither the variable on which the arrows converge, nor any of its descendants are observed. For instance, the path $A_4 \rightarrow A_8 \leftarrow A_5 \rightarrow A_6$ is blocked at A_8 only if variable A_8 is unobserved.

2.1.2 Inserting evidence

Beyond representing conditional independences between variables, Bayesian Networks provide a framework for inference and computation. Essentially, messages are exchanged through the graphical structure in order to compute the marginals of a set of variables given findings on another set. We will discuss the method for exact inference, which we employ in subsequent chapters of the thesis, in section 2.1.3. A more extensive overview of exact and approximate inference methods in probabilistic graphical models can be found in (Koller and Friedman, 2009)

Hard evidence. Evidence on a variable means that its value is observed. Such evidence is called "hard" evidence as it constrains a variable to a specific value. In a Bayesian Network over the universe $\mathcal{U}$ a hard finding can be represented by a unit vector potential $\boldsymbol{e}_i$, with a 1 at the i-th position to mark the state of an observed variable. This finding implies that certain states are impossible. If $\boldsymbol{e}$ groups all observations over all variables $\mathcal{U}$, then the joint probability over $\mathcal{U}$ including finding $\boldsymbol{e}$ is:

$$p(\mathcal{U}, e) = p(\mathcal{U}) \cdot \boldsymbol{e} = \prod p(A_i | parents(A_i)) \cdot \prod_{j=1}^{M} \boldsymbol{e}_j \tag{2.2}$$

From the joint probability $p(\mathcal{U}, e)$ one can calculate the conditional probability of any variable A given evidence e as:

$$p(A|e) = \frac{p(A, e)}{p(e)} = \frac{\sum_{\mathcal{U} \setminus A} p(\mathcal{U}, e)}{\sum_{\mathcal{U}} p(\mathcal{U}, e)} \tag{2.3}$$

Virtual evidence. Sometimes evidence about a variable is uncertain, such as the observation "the car in the far distance was white with 80% chance, but it might also have been grey". In the framework of Bayesian Networks it is also possible to deal with uncertain evidence. One type of uncertain evidence is called *virtual evidence* (Pearl, 1988), which conveys relative likelihood ratios of one event against another. In the previous example the likelihood ratio white:grey is $4:1$. *Virtual evidence* represents the belief in the true state of a variable by an agent, which is independent of previous information or other hypotheses in the system. Further, *virtual evidence* statements might be improved and updated by reasoning about other variables in the Bayesian Network. Graphically, *virtual evidence* is represented as an observed node VE (virtual evidence node) with a single parent X, which corresponds to the variable whose value is uncertain. The uncertain statement about variable X is reflected then in the probability table, $p(VE \mid X)$, which is added to the original Bayesian Network. Any inference algorithm on the Bayesian Network can refine the distribution on variable X by combining knowledge about other variables in the system with the uncertain statement $p(VE \mid X) = (p(VE|x_1), \ldots, p(VE|x_n)))$.

Soft evidence. Another type of uncertain evidence is called *soft evidence* (Valtorta et al., 2002). *Soft evidence* differs from *virtual evidence* in the sense that no later observations can improve knowledge about a certain statement. The agent who conveys *soft evidence* is authoritative about her probabilistic statement. Soft evidence includes observations about the marginal distribution of one or several variables, conditional probability distributions, probability assignments to arbitrary logical formulae or probability assignments to probabilistic functions. All types of soft evidence are modelled by an observation node SE, with parents including all variables involved in the statement. The probability tables, $p(SE|parents(SE))$ and states of the soft evidence node SE differ depending on the type of uncertain statement. Unfortunately, the general inference algorithms for Bayesian Networks cannot include soft evidence, since the inference mechanism might change the probability distributions of the statements involving soft evidence. (Kim et al., 2004) propose an extension of the junction-tree algorithm described in the following section, which can deal with soft evidence.

2.1.3 Exact Inference: The Junction-Tree Algorithm

In order to describe the *junction tree algorithm* (Jensen and Nielsen, 2007) it is necessary to introduce some graph theoretic notation and properties.

Let us consider a set of potentials, $\mathbf{\Phi} = \{\phi_1, \cdots, \phi_m\}$ over a collection of discrete variables $\mathcal{U} = \{A_1, \cdots, A_n\}$ with $domain(\phi_i) = \mathcal{D}_i$. The *domain graph* for $\mathbf{\Phi}$ is an undirected graph with the nodes in $\mathcal{U}$ and pairwise links between all variables inside a common domain. The domain graph of a Bayesian Network contains all nodes of the network with the same links as the original network. Additionally,

pairwise links are added between all nodes sharing a common child. For this reason the domain graph of a Bayesian Network is often called a *moral graph*.

Variable elimination. The next algorithm details how to eliminate a variable X from a set of potentials $\mathbf{\Phi}$. This algorithm proceeds in three stages. First it

Algorithm 1: Eliminate variable X from a set of potentials $\mathbf{\Phi}$

Data: variable X and set of potentials $\mathbf{\Phi}$
1 $\mathbf{\Phi}_X = \{\phi_i \in \mathbf{\Phi} \mid X \in domain(\phi_i)\}$ `potentials with` X `in their domain.`
2 $\phi^{-X} = \sum_X \prod \mathbf{\Phi}_X$ `product of` $\mathbf{\Phi}_X$ `marginalising out` X.
3 $\mathbf{\Phi}^{-X} = \{\mathbf{\Phi} \setminus \mathbf{\Phi}_X, \phi^{-X}\}$ `new potential set without` X `in its domain.`
4 **return** *set of potentials* $\mathbf{\Phi}^{-X}$ *without variable* X *in their domains.*

creates a new set $\mathbf{\Phi}_X$, which contains all potentials of $\mathbf{\Phi}$ with variable X in their domain (line 1). Then, it marginalises out variable X from the set $\mathbf{\Phi}_X$ (line 2). Marginalising out in the calculus of potentials means to sum over all discrete values of variable X once the potentials including variable X have been multiplied, i.e. $\phi^{-X} = \sum_X \prod \mathbf{\Phi}_X$. The last step in line 3, adds the potential ϕ^{-X}, which no longer contains X in its domain, to the remaining set $\mathbf{\Phi} \setminus \mathbf{\Phi}_X$. After eliminating variable X the resulting set is $\mathbf{\Phi}^{-X} = \{\mathbf{\Phi} \setminus \mathbf{\Phi}_X, \phi^{-X}\}$. In the domain graph of $\mathbf{\Phi}^{-X}$ all neighbours of variable X are now pairwise linked, since the new term ϕ^{-X} has a domain including all neighbours of X.

It still remains to establish an elimination order in order to obtain the marginal of all variables A_i, that is $\phi_{A_i} = (\prod \mathbf{\Phi})^{\downarrow A_i} = \sum_{\mathcal{U} \setminus A_i} \prod \mathbf{\Phi}$. We have seen in the previous algorithm that eliminating a variable X results in a new domain graph with all neighbours of X pairwise linked. A *perfect elimination sequence* does not introduce any additional links in the original domain graph, which is more efficient in terms of storage and computation.

Terminology. Every elimination sequence produces a set of potentials with associated domains. The *domain set* of an elimination sequence contains all set of potentials generated during the elimination, without considering subsets. Unfortunately, even a perfect elimination sequence does not always guarantee that the resulting domain set will contain only the original domains. However, all *perfect elimination sequences* produce the same domain set, which corresponds to the set of *cliques* of the original domain graph. A *clique* in the domain graph is a set of fully connected nodes, which is no subset of any larger fully connected set. Fully connect sets are called *complete sets*. Thus, a *clique* is a *maximal complete set*. Further, nodes adjacent to a neighbouring complete set are called *simplicial nodes*. A simplicial node and its neighbourhood form a clique. Simplicial nodes play an

important role in perfect elimination sequences. An undirected graph has a perfect elimination sequence if and only if all nodes can be eliminated by successively eliminating *simplicial nodes*. Such undirected graphs with perfect elimination sequences are called *triangulated graphs*. Every triangulated graph has at least two non-adjacent simplicial nodes, and whenever a simplicial node is eliminated, the resulting graph is again triangulated.

Join-trees. In the *junction tree algorithm* the notion of *join trees* is also important. A *join tree* is a particular organisation of the cliques of an undirected graph into a tree structure. For every pair of nodes V and W in the join tree, all nodes in the path from V to W contain the intersection $V \cap W$. An undirected graph is *triangulated*, i.e. it contains a perfect elimination sequence, if and only if its cliques can be organised into a *join tree*. A join tree is constructed from a triangulated graph according to algorithm 2.

Algorithm 2: Construct join-tree from undirected triangulated graph

Data: undirected triangulated graph G
$G' := G$
cliques:= $\varnothing$
join-tree := $\varnothing$
i,lastId := 0
while *nodes remain in* G' **do**
  sn := search-simplicial-node(G') : `at least 2 per triang. graph`
  CLSN := get-clique(sn) : `neighbours of` *sn* `form clique`
  [ElimNodes,RestNodes] := eliminate-nodes-only-in-clique(CLSN)
  `Fill the ''join-tree'' data structure`
  join-tree[i].separator := RestNodes : `remaining nodes`
  join-tree[i].clique := CLSN : `current clique`
  join-tree[i].id := lastId + ♯(ElimNodes) : `new identifier`
  join-tree[i].parent := $\varnothing$
  i = i+1 , lastId := join-tree[i].id
  $G' = G' \setminus$ ElimNodes
forall *elements of join-tree* **do**
  join-tree[i].parent = search-parent(i,join-tree) `j=parent(i)` $\Leftrightarrow$ `i<j` $\wedge$ `join-tree[i].separator` $\subset$ `join-tree[j].clique`
return *join-tree*

The join-tree construction proceeds by sequentially eliminating simplicial nodes from a triangulated graph. This is always possible as a triangulated graph has at least two non-adjacent simplicial nodes. Further, the neighbouring nodes and the simplicial nodes together form a clique. This clique contains two kind of nodes, those with neighbours only in the clique and those whose neighbourhood extends beyond the clique. The latter kind of nodes are connected to other nodes, which do not share links with all members of the clique. Eliminating any of these nodes

would add links to the domain graph. Therefore, the first nodes to be eliminated are those with only internal connections in the clique under consideration. The remaining nodes form the *separator* which divides the graph into two parts. All paths connecting the two parts of the graph necessarily traverse this separator. In case the join-tree is constructed from a Bayesian Network, the two parts are d-separated given the separator. Each clique and separator are numbered according to the cardinality of the set of eliminated nodes (line 12).

The elimination process continues as long as nodes remain in the triangulated graph. The second part of the algorithm, starting in line 16, creates a tree structure by connecting separators to higher level cliques, which contain each separator completely. This is always possible because the separators are complete sets, whose nodes are eliminated when dealing with higher-level cliques. The structure is a tree since each separator has at most one parent. It also fulfils the join-tree condition. A node which is member of two different cliques belongs to all cliques along the path connecting them.

A junction-tree for a triangulated domain graph of a set of potentials $\mathbf{\Phi} = \{\phi_1, \cdots, \phi_n\}$ is a join-tree with two additions. Firstly, each potential $\phi_i \in \mathbf{\Phi}$ is associated to the clique of the join tree which fully contains its domain. Potentials are only associated to one clique at a time. Secondly, the separators which interconnect cliques in the junction-tree maintain two mailboxes, one outgoing and another one incoming.

Let us consider one clique V of the *junction-tree* with associated set of potentials, $\mathbf{\Phi}_V$, and which is connected to separators $S_1, \cdots, S_k$ and S. V can send a message to separator S whenever it has received messages from all remaining separators. The outgoing message to separator S is ψ^S, which contains the union of all incoming potentials and the clique potential set, $\mathbf{\Phi}_V$. This union of potentials is projected down to the variables in the separator set S, i.e. all variables except those in the separator domain are marginalised out. The potentials forming the outgoing message ψ^S can be divided into two subsets, namely all potentials whose domain is fully contained in separator S, $\psi^S_{dom \subseteq S}$, and those with a domain extending outside S, i.e. $\psi^s_{dom \cap (\mathcal{U} \setminus S) \neq \varnothing}$. We summarise the structure of the outgoing message ψ^S in the next set of equations.

$$\psi^S = (\mathbf{\Phi}_V \cup \psi_1 \cup \cdots \cup \psi_k)^{\downarrow S} \tag{2.4a}$$

$$\psi^S_{domain \subseteq S} = \{\psi \in \{\mathbf{\Phi}_V, \psi_1, \cdots, \psi_k\} \mid domain(\psi) \subseteq S\} \tag{2.4b}$$

$$\psi^s_{dom \cap (\mathcal{U} \setminus S) \neq \varnothing} = \{\psi \in \{\mathbf{\Phi}_V, \psi_1, \cdots, \psi_k\} \mid domain(\psi) \cap (\mathcal{U} \setminus S) \neq \varnothing\} \tag{2.4c}$$

Separating the potentials of ψ_S according to their domains allows to apply the variable elimination algorithm (alg. 1) to the outgoing message. Here it is necessary

to eliminate all variables except those contained in the domain S, i.e. the elimination set in algorithm 1 is $X = \mathcal{U} \setminus S$.

$$\psi^S = \left(\psi^S_{domain \subseteq S} \cup \psi^s_{dom \cap (\mathcal{U} \setminus S) \neq \varnothing}\right)^{\downarrow S} \tag{2.5a}$$

$$= \left\{\psi^S_{domain \subseteq S}, \left(\prod \psi^s_{dom \cap (\mathcal{U} \setminus S) \neq \varnothing}\right)^{\downarrow S}\right\} \tag{2.5b}$$

$$= \left\{\psi^S_{domain \subseteq S}, \sum_{\mathcal{U} \setminus S} \prod \psi^s_{dom \cap (\mathcal{U} \setminus S) \neq \varnothing}\right\} \tag{2.5c}$$

The main idea behind the *junction-tree* is to define a message distribution schedule, which allows to calculate the marginal of *any* variable easily. It proceeds in two stages, the *collect evidence* stage and the *distribution* stage. Collecting evidence means sending messages from the leaf nodes in the junction tree to an arbitrary root node. The leaf nodes first send messages to their separators according to the protocol of equation 2.5. Sending these messages corresponds to eliminating all simplicial nodes which are absent in the parent clique. Whenever a clique of the second level receives potentials from all incoming separators it can forward another message to its upper level. The process continues until all evidence reaches the root node. This process corresponds to a perfect elimination sequence finishing with the variables of the root clique.

Once the root node has collected all evidence, it initiates the *distribution* stage by sending messages to its outgoing separators. Each separator of the root clique receives a message according to equation 2.5. This is possible since the incoming messages of the root node have been calculated in the *collect evidence* phase. The process stops when evidence reaches the leaf nodes of the junction tree. At this point all mailboxes of all separators in the tree are full with outgoing and an incoming messages.

In order to calculate the marginal of any variable, one multiplies all incoming messages with the clique potentials of the clique which contains the variable of interest. Then it is just necessary to eliminate all remaining variables according to algorithm 1. Alternatively, if the variable of interest is contained within a separator, it is also possible to obtain the marginal of this variable by multiplying incoming and outgoing messages of the separator and marginalising out the remaining variables.

2.1.4 Dynamic Bayesian Networks

A Dynamic Bayesian Network (DBN) extends basic Bayesian Networks to sequences over semi-infinite random variables. For instance, repeating the topology of figure 2.1 indefinitely would create a Dynamic Bayesian Network. The adjective "dynamic" refers to the underlying discrete-time dynamical system, which is modelled by the Dynamic Bayesian Network, and not to changes in the network's structure. Dynamic Bayesian Networks are indexed by a discrete variable t that increase each time a new observation arrives.

(Murphy, 2002) defines Dynamic Bayesian Networks as a pair $(BN_1, BN_{\rightarrow})$. BN_1 represents the prior Bayesian Network for the variables in the first time slice $\mathcal{U}_1$. $BN_{\rightarrow}$, on the other hand, is a two-slice temporal Bayesian Network, which models the transition probability $p(\mathcal{U}_t \mid \mathcal{U}_{t-1})$ by a directed acyclic graph and a factorisation according to:

$$p(\mathcal{U}_t \mid \mathcal{U}_{t-1}) = \prod_{i=1}^{N} p(A_t^i \mid parents(p(A_t^i))) \tag{2.6}$$

A_t^i is the $i-th$ node at time t and $parents(p(A_t^i)$ is the set of parent nodes immediately preceding A_t^i. Parent nodes can belong either to the same slice or to previous time slices. Equation 2.6 just considers first-order transitions for notational simplicity, but the formalism generalises to higher-order systems as well.

In the two-slice temporal Bayesian Network, $BN_{\rightarrow}$, only the probabilities in the second slice, $p(A_t^i \mid parents(p(A_t^i)))$ are relevant, the first slice is parameterless. The intra-slice topology of $BN_{\rightarrow}$ is arbitrary, as long as the directed acyclic graph condition is preserved. Further, directed arcs within a slice represent instantaneous causation, and nodes with edges across time slices to equivalent nodes are termed *persistent*. The parameters $p(A_t^i \mid parents(p(A_t^i)))$ are normally considered time-invariant, i.e. shared across all time slices. Time-varying parameters can be modelled by including them in the Dynamic Bayesian Network as random variables with associated conditional probability distributions. Also switching between discrete set of parameters is possible by adding a hidden discrete variable, which selects the parameter set of different regimes.

Unrolling T time slices of the Dynamic Bayesian Network $(BN_1, BN_{\rightarrow})$ leads to the following distribution:

$$p(\mathcal{U}_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} p(A_t^i \mid parents(p(A_t^i))) \tag{2.7}$$

The interface algorithm. (Murphy, 2002) proposes the *interface algorithm* for exact inference in Dynamic Bayesian Networks. This algorithm extends the junction-tree algorithm for static Bayesian Networks of section 2.1.3 to Dynamic Bayesian Networks. Inference proceeds on the reduced representation of the Dynamic Bayesian Network presented previously. The *interface algorithm* exploits a property of all nodes with outgoing arrows to the next time slice. (Murphy, 2002) calls such nodes the *forward interface*. They d-separate the past from the future, which allows to establish a information propagation schedule considering one time slice at a time.

Let us call $\mathcal{I}_{t-1}$ the set of nodes in slice $t-1$ with children in the next time slice and $\mathcal{U}_t$ the set of all nodes of slice t. The *interface algorithm* first constructs the $1\frac{1}{2}$-Dynamic Bayesian Network including nodes $\mathcal{H}_t = \mathcal{I}_{t-1} \cup \mathcal{U}_t$, with all remaining nodes in slice $t-1$ eliminated. The potentials of eliminated nodes have been propagated to $\mathcal{H}_t$.

The next step is to calculate a junction-tree $\mathcal{JT}_t$ for $\mathcal{H}_t$ with the requirement that the interfaces $\mathcal{I}_t$ and $\mathcal{I}_{t-1}$ belong to separate cliques. This requirement can be enforced by adding edges to the moral graph between all nodes of $\mathcal{I}_t$ and $\mathcal{I}_{t-1}$ respectively. It is necessary to create two separate cliques since the probabilities $p(\mathcal{I}_t)$ and $p(\mathcal{I}_{t-1})$ are both needed. The junction-trees of different time-slices are then connected together by their interfaces, as each interface is a separator in the global structure.

Inference proceeds by exchanging messages through the common separators in two sweeps, first forwards and then backwards. The forwards pass collects incoming evidence in $\mathcal{JT}_t$ to $\mathcal{I}_t$ from interface $\mathcal{I}_{t-1}$ as well as from all internal nodes of slice t. Once all evidence to $\mathcal{I}_t$ is collected, the process can be repeated in the next slice until reaching the end of a finite chain.

A backwards pass is also necessary in order to propagate messages to all nodes in every junction-tree. In the backward pass, evidence arrives from slice $t+1$ to the interface $\mathcal{I}_t$, which is the root node of junction-tree $\mathcal{JT}_t$. The root then distributes evidence to the leaves of $\mathcal{JT}_t$, including $\mathcal{I}_{t-1}$, and the process can continue in the previous junction tree, $\mathcal{JT}_{t-1}$.

This procedure corresponds to calculating the *smoothing* distribution. In order to compute the *filtering* distribution, the backward pass includes only the current slice.

2.2 Gaussian Processes

This section briefly introduces Gaussian Processes based on (Rasmussen and Williams, 2006).

2.2.1 Model description

A Gaussian Process ($\mathcal{GP}$) defines a distribution over functions $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ in function space. Function values, $f(\boldsymbol{x})$, form an infinitely uncountable collection of random variables, with joint Gaussian distribution of every finite subset of them (Rasmussen and Williams, 2006).

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x\prime})\right) \tag{2.8a}$$

$$m(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right];$$

$$k(\boldsymbol{x}, \boldsymbol{x\prime}) = \mathbb{E}\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f(\boldsymbol{x\prime}) - m(\boldsymbol{x\prime})\right)\right] \tag{2.8b}$$

where $\sim$ denotes distributed according to. Therefore, a Gaussian Process is completely specified by the mean, $m(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$, and the covariance function $k(\boldsymbol{x}, \boldsymbol{x\prime}) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Both mean and covariance functions are defined in function space but can be calculated directly in input space.

In order to simplify the notation, the mean function can be set to zero, provided that all training values are normalized to zero mean. Therefore, the covariance function encodes all the properties about the unknown function to be learned, $f(\boldsymbol{x})$. Not every function is a valid covariance function. Only positive semi-definite covariance functions induce a Gaussian Process (Rasmussen and Williams, 2006).

Covariance functions. The covariance functions, which are employed for user modelling in chapter 4 are the squared exponential (SE) and the rational quadratic (RQ):

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x\prime}) = \sigma_f \cdot \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x\prime})^T \boldsymbol{P}^{-1}(\boldsymbol{x} - \boldsymbol{x\prime})\right) \tag{2.9a}$$

$$k_{RQ}(\boldsymbol{x}, \boldsymbol{x\prime}) = \sigma_f \cdot \left[1 + \frac{(\boldsymbol{x} - \boldsymbol{x\prime})^T \boldsymbol{P}^{-1}(\boldsymbol{x} - \boldsymbol{x\prime})}{2\alpha}\right]^{-\alpha} \tag{2.9b}$$

where σ_f is a signal strength parameter and $\boldsymbol{P}$ is a diagonal matrix of inverse length scales. This matrix weighs the relative importance of each input dimension in the covariance function. $\alpha > 0$ is a positive real number. All these parameters can be estimated implicitly for a Gaussian Process out of a discrete set of noisy

observations $(\boldsymbol{X}, \boldsymbol{y}) = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where unobserved function values, $f(\boldsymbol{x}_i)$, are corrupted by additive independent identically distributed Gaussian noise of variance σ_n^2, i.e. $y_i = f(\boldsymbol{x}_i) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Predictions. The goal of Gaussian Process regression is to predict unseen function values $\boldsymbol{f}_*$ for a collection of test points $\boldsymbol{X}_* = [\boldsymbol{x}_*^1 \ldots \boldsymbol{x}_*^M]$, i.e. $\boldsymbol{f}_* = [f(\boldsymbol{x}_*^1) \ldots f(\boldsymbol{x}_*^M)]^T$. Since the unseen function values $\boldsymbol{f}_*$ and the observed targets $\boldsymbol{y}$ are distributed according to a Gaussian Process, they have a joint Gaussian distribution with the following properties:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K_{\boldsymbol{y}}(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}, \boldsymbol{X}_*) \\ K(\boldsymbol{X}_*, \boldsymbol{X}) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right) \tag{2.10}$$

The predictive distribution $p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*)$ conditions the joint prior distribution of equation 2.10 on the training data $(\boldsymbol{X}, \boldsymbol{y})$ and the test inputs $\boldsymbol{X}_*$.

$$\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_* \sim \mathcal{N}\left(\bar{\boldsymbol{f}}_*, cov(\boldsymbol{f}_*)\right) \tag{2.11a}$$

$$\bar{\boldsymbol{f}}_* = K(\boldsymbol{X}_*, \boldsymbol{X}) K_{\boldsymbol{y}}^{-1} \boldsymbol{y} \tag{2.11b}$$

$$cov(\boldsymbol{f}_*) = K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K(\boldsymbol{X}_*, \boldsymbol{X}) K_{\boldsymbol{y}}^{-1} K(\boldsymbol{X}, \boldsymbol{X}_*)$$

$$K_{\boldsymbol{y}} = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}_{N \times N} \tag{2.11c}$$

where eq. 2.11a states the predictive distribution of unknown function values conditioned on the observations, eq. 2.11b defines the mean and eq. 2.11c the variance of the predictive distribution. $K(\boldsymbol{X}, \boldsymbol{X})$ is the symmetric, positive definite *Gramm* matrix of training samples $\boldsymbol{x}$, i.e. a matrix where each entry corresponds to the covariance function evaluated between any two training samples. Likewise, $K(\boldsymbol{X}_*, \boldsymbol{X})$ is the covariance matrix between training and test samples and $K(\boldsymbol{X}, \boldsymbol{X}_*)$ its transpose. $\boldsymbol{I}_{N \times N}$ is a square identity matrix of size $N \times N$ and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the multivariate normal probability density function.

Computation. A numerically efficient and robust algorithm to calculate the predictive distribution in eq. 2.11 is proposed in (Rasmussen and Williams, 2006) and is restated in alg. 3 for the sake of completeness.

$K_{\boldsymbol{y}}$ is positive semi-definite and can be represented as a product of a lower triangular matrix $\boldsymbol{L}$ and its transpose $\boldsymbol{L}^T$. $\boldsymbol{L}$ is called the Cholesky factor[1] of the matrix

[1]The Cholesky factor is useful for solving linear systems of equations of the form $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a positive semi-definite matrix. The associative property of matrix multiplication

Algorithm 3: Numerically robust predictive distribution

Data: $\boldsymbol{X}$ (training set), $\boldsymbol{y}$ (training targets), $k(\cdot,\cdot)$ (covariance), σ_n^2 (noise), $\boldsymbol{X}_*$ (predictions)

1 $\boldsymbol{L} := \texttt{Cholesky}(K(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}_{N\times N})$

2 $\boldsymbol{\alpha} := \boldsymbol{L}^T \backslash (\boldsymbol{L} \backslash \boldsymbol{y})$

3 **forall** $\boldsymbol{x}_* \in \boldsymbol{X}_*$ **do**

4 $\quad \boldsymbol{k}_* = K(\boldsymbol{X}, \boldsymbol{x}_*)$

5 $\quad \bar{f}_* = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_*) = \boldsymbol{k}_*^T \boldsymbol{\alpha}$

6 $\quad \boldsymbol{\psi} := \boldsymbol{L} \backslash \boldsymbol{k}_*$

7 $\quad \mathbb{V}[y_*] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{\psi}^T \boldsymbol{\psi} + \sigma_n^2$

8 $p(\boldsymbol{y} \mid \boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{N}{2} \log 2\pi$

9 **return** $\bar{y}_* = \bar{f}_*$ *(mean prediction),* $\mathbb{V}[y_*]$ *(variance),* $p(\boldsymbol{y} \mid \boldsymbol{X})$ *(marginal likelihood training)*

and can be computed in $O(N^3/6)$ time with numerically robust algorithms. The "\" operator in lines 2 and 6 of alg. 3 denotes the operation of solving a linear system of equations $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{s}$ for vector $\boldsymbol{z}$, i.e. $\boldsymbol{z} = \boldsymbol{A} \backslash \boldsymbol{s}$, which is straightforward for triangular matrices. In line 2, $\boldsymbol{\alpha}$ is the component of the predictive mean, which does not depend on the input, $\boldsymbol{x}_*$, and verifies $K_{\boldsymbol{y}}\boldsymbol{\alpha} = \boldsymbol{L}\boldsymbol{L}^T\boldsymbol{\alpha} = \boldsymbol{y}$. Inverting $K_{\boldsymbol{y}}$ can then be done by solving first $\boldsymbol{L}\boldsymbol{b} = \boldsymbol{y}$ for $\boldsymbol{b}$ and afterwards $\boldsymbol{L}^T\boldsymbol{\alpha} = \boldsymbol{b}$ for $\boldsymbol{\alpha}$ (cf. lines 1- 2 of alg. 3 and footnote 1).

Conversely, vector $\boldsymbol{\psi}$ in line 6 must be computed for every new input with the Cholesky factor of $K_{\boldsymbol{y}}$, since $\boldsymbol{k}_*^T K_{\boldsymbol{y}}^{-1} \boldsymbol{k}_* = \boldsymbol{k}_*^T (\boldsymbol{L}\boldsymbol{L}^T)^{-1} \boldsymbol{k}_*$ and consequently, $\boldsymbol{\psi} = \boldsymbol{L}^{-1}\boldsymbol{k}_* \Leftrightarrow \boldsymbol{\psi} = \boldsymbol{L} \backslash \boldsymbol{k}_*$. The Cholesky decomposition in line 1 can be performed offline, after all training data have been gathered. However, in order to obtain the predictive distribution for a new input, lines 4 to 7 have to be executed online, which involve a computational cost of $O(N^2/2)$, due to the calculation of $\boldsymbol{\psi}$. One such vector $\boldsymbol{\psi}(\boldsymbol{i}_k)$ must be computed per user plan hypothesis in our probabilistic framework for plan recognition.

2.2.2 Bayesian model selection

It follows a brief introduction to Bayesian model selection according to (Rasmussen and Williams, 2006).

Models $\mathcal{M}$ normally have a hierarchical structure. At the lowest layer are the parameters of the model $\boldsymbol{w}$ such as the weights in a neural network. In turn, a

allows to reformulate the linear equation system as the following two sub-problems: $\boldsymbol{A}\boldsymbol{z} = (\boldsymbol{L}\boldsymbol{L}^T)\boldsymbol{z} = \boldsymbol{L}(\boldsymbol{L}^T\boldsymbol{z}) = \boldsymbol{L}\boldsymbol{s} = \boldsymbol{b}$, with $\boldsymbol{L}^T\boldsymbol{z} = \boldsymbol{s}$. Then, $\boldsymbol{L}\boldsymbol{s} = \boldsymbol{b}$ is solved for $\boldsymbol{s}$ by forward substitution and $\boldsymbol{L}^T\boldsymbol{z} = \boldsymbol{s}$ is solved for $\boldsymbol{z}$ via backward substitution. Furthermore, the determinant of $\boldsymbol{A}$ is $|\boldsymbol{A}| = \prod_{i=1}^{N} L_{ii}^2$, where L_{ii} are the diagonal elements of $\boldsymbol{L}$.

set of hyper-parameters $\boldsymbol{\theta}$ control the distribution of model parameters $\boldsymbol{w}$. At the highest level the designer has the choice among a discrete set of models $\mathcal{M}_i$.

In Bayesian learning one is interested in the posterior over model parameters $\boldsymbol{w}$ given the training samples $\boldsymbol{X}$ and target values $\boldsymbol{y}$ for a specific instance of the model class $\mathcal{M}_i$ defined by the hyper-parameters $\boldsymbol{\theta}$:

$$\text{param. posterior: } p\big(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big) = \frac{p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \mathcal{M}_i\big) \cdot p\big(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{M}_i\big)}{p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big)} \quad (2.12)$$

where $p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \mathcal{M}_i\big)$ is the data likelihood and $p\big(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{M}_i\big)$ is a parameter prior. This prior encodes the prior knowledge about the distribution of the model parameters without considering any data. The normaliser $p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big)$ is called *marginal likelihood*. The marginal likelihood is independent of the parameters $\boldsymbol{w}$ because they have been integrated (marginalised) out.

$$\text{marg. likelihood: } p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big) = \int p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \mathcal{M}_i\big) \cdot p\big(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{M}_i\big) d\boldsymbol{w} \quad (2.13)$$

At the second level we define the posterior over hyper-parameters, where the marginal likelihood in equation 2.13 represents the data likelihood of the hyper-parameters.

$$\text{hyper-posterior: } p\big(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{X}, \mathcal{M}_i\big) = \frac{p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big) \cdot p\big(\boldsymbol{\theta} \mid \mathcal{M}_i\big)}{p\big(\boldsymbol{y} \mid \boldsymbol{X}, \mathcal{M}_i\big)} \quad (2.14)$$

where $p\big(\boldsymbol{\theta} \mid \mathcal{M}_i\big)$ is a hyper-prior for model class $\mathcal{M}_i$. The normalising constant $p\big(\boldsymbol{y} \mid \boldsymbol{X}, \mathcal{M}_i\big)$ can be calculated by integrating the denominator of equation 2.14.

$$p\big(\boldsymbol{y} \mid \boldsymbol{X}, \mathcal{M}_i\big) = \int p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big) \cdot p\big(\boldsymbol{\theta} \mid \mathcal{M}_i\big) d\boldsymbol{\theta} \quad (2.15)$$

At the top of the hierarchy we can compute the posterior probability for the model class $\mathcal{M}_i$, and can thus compare different types of models for a given dataset.

$$\text{model posterior: } p\big(\mathcal{M}_i \mid \boldsymbol{y}, \boldsymbol{X}\big) = \frac{p\big(\boldsymbol{y} \mid \boldsymbol{X}, \mathcal{M}_i\big) \cdot p\big(\mathcal{M}_i\big)}{p\big(\boldsymbol{y} \mid \boldsymbol{X}\big)} \quad (2.16)$$

The normaliser in the hyper-posterior (equation 2.14) plays the role of model likelihood in the calculation of the posterior over models in equation 2.16. In order

to normalise the posterior over models, it is necessary to sum the denominator in equation 2.16 over all plausible models.

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \sum_i p(\boldsymbol{y} \mid \boldsymbol{X}, \mathcal{M}_i) \cdot p(\mathcal{M}_i) \tag{2.17}$$

The prior over models $p(\mathcal{M}_i)$ is normally flat in absence of a preference for a particular model. Then the posterior over models is proportional to equation 2.15, which defines an integral over the space of hyper-parameters $\boldsymbol{\theta}$. Generally, this integral is difficult to evaluate, which renders the posterior over hyper-parameters (equation 2.14) unusable.

A common approximation is to maximise the marginal likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i)$ (equation 2.13) with respect to the hyper-parameters $\boldsymbol{\theta}$, which is known as type-II maximum likelihood (ML-II). Unfortunately, such optimisation might give rise to over-fitting, especially if the number of hyper-parameters is large. The integral in equation 2.15 can then be approximated with a local expansion around the maximum of the marginal likelihood. The quality of the approximation is better for peaked hyper-posteriors (equation 2.14). Generally, the hyper-posterior is more peaked than the posterior over parameters (Rasmussen and Williams, 2006).

Comparing models in terms of the marginal likelihood automatically enforces a trade-off between model fit and model complexity. The marginal likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i)$ is a probability distribution over all possible datasets $\boldsymbol{y} \in \mathcal{Y}$, i.e. it is normalised over the space $\mathcal{Y}$. As figure 2.2 shows, a complex model that can explain many possible datasets must lead to a low marginal likelihood. Conversely, a simple model has a high marginal likelihood for a reduced number of datasets. Therefore, the marginal likelihood favours the least complex model that can still explain the observed data.

Automatic relevance determination

Normally, the formalism explained in the previous section cannot be applied directly to most machine learning algorithms because the necessary integrals are intractable. Gaussian Process are an exception to this rule and the marginal likelihood has a closed form solution. Hence, Bayesian Model selection with the maximum likelihood type-II approximation becomes possible.

Gaussian Process Regression is a non-parametric learning technique where the data represent the model. In this case, the parameters $\boldsymbol{w}$ are the latent function values at the training inputs, i.e. $\boldsymbol{f}(\boldsymbol{x}) = [f(\boldsymbol{x}_1) \dots f(\boldsymbol{x}_N)]^T$. The marginal likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i)$ for a Gaussian Process has the following expression, where the

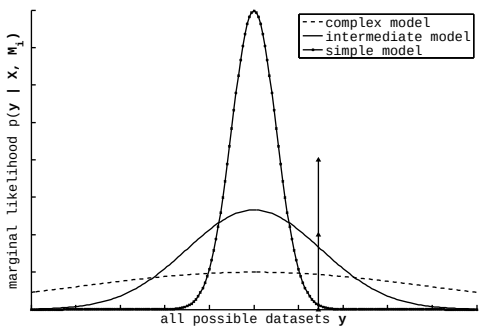


FIGURE 2.2: The marginal likelihood is the probability of the data $\boldsymbol{y}$ given the model $\mathcal{M}_i$. In the figure we represent three marginal likelihood distributions over an idealised representation of all possible datasets. Each marginal likelihood is a probability distribution over the data, i.e. it is normalised over the data $\boldsymbol{y}$. Therefore, for a given dataset marked by a vertical line, the marginal likelihood favours the least complex model that can explain the data. This figure is after (Rasmussen and Williams, 2006).

parameters $\boldsymbol{f}$ have been integrated out:

$$\log p\big(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{M}_i\big) = \underbrace{-\frac{1}{2}\boldsymbol{y}^T K_{\boldsymbol{y}} \boldsymbol{y}}_{\text{data fit}} \quad \underbrace{-\frac{1}{2}\log|K_{\boldsymbol{y}}|}_{\substack{\text{negative} \\ \text{complexity penalty}}} \quad \underbrace{-\frac{N}{2}\log 2\pi}_{\text{normaliser}} \tag{2.18}$$

where $K_{\boldsymbol{y}} = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n \boldsymbol{I}_{N\times N}$ is the Gramm matrix for the target values $\boldsymbol{y}$, and $|K_{\boldsymbol{y}}|$ its determinant. The hyper-parameters $\boldsymbol{\theta}$ correspond to the parameters of the covariance function of the model class $\mathcal{M}_i$.

Equation 2.18 combines three terms with a clear interpretation. Only the first term, the data-fit term $-\frac{1}{2}\boldsymbol{y}^T K_{\boldsymbol{y}} \boldsymbol{y}$, depends on the observed target values $\boldsymbol{y}$. The data-fit term penalises less flexible models that cannot explain the data $\boldsymbol{y}$. The second term, the complexity penalty $\frac{1}{2}\log|K_{\boldsymbol{y}}|$, is better for simpler models. Conversely, the negative complexity penalty decreases if the model is more complex. The third term is a normalisation constant proportional to the size of the dataset.

The optimal hyper-parameters $\boldsymbol{\theta}_{ML-II}$ trade-off model complexity and data-fit. They favour the least complex model, which fits the data. This optimum combines the highest negative complexity penalty with the best data fit term. Further, different model classes $\mathcal{M}_i$ can be compared in terms of their marginal likelihood for their optimal set of hyper-parameters $\boldsymbol{\theta}_{ML-II}$.

There is no guarantee for the absence of multiple local optima in the marginal likelihood. Each local maximum is a particular interpretation of the data. This problem is especially acute for small datasets, which cannot rule out different interpretations confidently. For larger datasets, one maximum is orders of magnitude larger than the rest. In general, more data leads to a more peaked marginal likelihood. However, it is necessary to distinguish two situations concerning the size of the data. On the one hand, fixed-domain asymptotics means an increasing density of observations of a region. On the other hand, additional data can correspond to an increasing region, which is called increasing-domain asymptotics. The latter is the case for time-series and additional data does not necessarily imply a better estimation of the hyper-parameters (Rasmussen and Williams, 2006).

In the remainder of the text we will refer to the estimation of the hyper-parameters of the Gaussian Process as automatic relevance determination (ARD). ARD produces the relative weights of the inputs (the length scale matrix $\boldsymbol{P}$) in the computation of the user model. Hence, it enables to analyse which input dimensions are significant for a particular user when steering a wheelchair. Further, we will omit the dependency of the model class $\mathcal{M}_i$ and the hyper-parameters $\boldsymbol{\theta}$ from the marginal likelihood and simply write $p\left(\boldsymbol{y}|\boldsymbol{X}\right)$.

2.2.3 Sparse Gaussian Processes

(Csato and Opper, 2002) propose an approximation, which allows the practical application of Gaussian Processes in our probabilistic framework for plan recognition.

(Csato and Opper, 2002) calculate a sparse[2] online approximation to the full Gaussian Process posterior. Each training sample is presented sequentially during the training phase. Since training the Gaussian Process with the new sample involves a Bayesian update, the resulting posterior does not necessarily have to be Gaussian, depending on the data likelihood. In order to maintain Gaussianity a projection to the closest Gaussian distribution, $\hat{p}$, is performed. Closeness of distributions is measured by the Kullback-Leibler divergence $KL(p_{post}|\hat{p}) = \int p_{post}(z) \ln \frac{p_{post}(z)}{\hat{p}(z)} dz$. Minimizing this projection results in matching the first two moments of p_{post} and $\hat{p}$ (Opper, 1998). The online approximation is represented then as a series of coefficients and matrices, which are updated sequentially, and other coefficients and a matrix, which do not depend on the data. The sparse approximation to the Gaussian Process decides at each stage of the training procedure, whether to add a new training point to a set of basis vectors, or to keep the set of basis vectors constant and just update the weights. With suitable enlargement and deletion procedures, the set of basis vectors and all matrices involved in the Gaussian Process computation can be kept of a fixed size.

[2]Sparseness refers to the number of data examples, which define the model.

Chapter 3

Plan Recognition

> *"[...] we usually know how things will end, how they will evolve and what awaits us, where things are going and what their conclusion will be; everything is there on view, in fact, everything is visible very early on [...] you just have to look to see it, [...] there are not that many possible variations, the signs rarely deceive if we know how to read their meanings, if you are prepared to do so - but it is difficult and can prove catastrophic; one day [...] you detect hidden intentions that are never entirely hidden, or unconscious intentions before they become conscious to the person who should be concealing them, sometimes you foresee what someone will do before that person has foreseen or known or even become aware of what this will be ..."*
>
> Javier Marías (⋆1951)
> *Your face tomorrow. Vol. I: Fever and Spear*, page 35 (Marías, 2007).

This chapter establishes a probabilistic framework to recognise the navigation plans of a user when interacting with a robot. The problem is "difficult" and can prove "catastrophic" if done incorrectly. A catastrophe could happen if human and robot misunderstand each other, a situation called *mode confusion*. Our goal is to avoid mode confusion as far as possible and to achieve instead a robot, which understands its user and which is therefore intuitive to control.

The chapter is structured as follows. First, we will briefly review the problem of plan recognition in other domains in order to set the context. Next, section 3.2 defines human-robot navigation in probabilistic terms. A robot interacting with a human is moving through potentially unknown environments, in which it needs to localise itself. Further, the human operator does not communicate her intentions explicitly. She interacts with the robot through her particular user input device, which might only provide a limited amount of information about her true plans. Therefore, the robot needs to recognise the plan of its user in addition to localising

itself in an unknown map. In order to address the robot's state estimation problem, we decompose the estimation into two sub-tasks, Simultaneous Localisation and Mapping (SLAM) and plan recognition. After a brief review of the SLAM problem (section 3.2.1), we describe a probabilistic framework for plan recognition in section 3.2.2. This framework proposes a representation of user plans that differs from previous approaches, which nearly all represent intentions in terms of specific assistance algorithms, e.g. *follow-corridor* or *dock-at-table*. We discuss the generation of global and local plans in section 3.2.3. Temporal reasoning of plan recognition with global plans is a direct application of the general plan recognition algorithm, which we describe in section 3.2.2. However, local plan recognition requires a novel approach to propagate information through time, which we introduce in section 3.2.4. The chapter concludes with an experimental section, where we describe plan recognition results for a discrete and a continuous robotic interface.

3.1 General description of plan recognition

The field of plan recognition can be traced back to early work by (Schmidt et al., 1978) on action understanding. Schmidt et al. define plan recognition as identifying the goals of an agent together with the plan to achieve those goals. The purpose is to organise observed actions into a plan structure. In the process, alternative hypotheses remain generic until incoming observations provide sufficient evidence to make plans more specific. For instance, somebody observing a person walking to the kitchen might postulate that this person wants to cook or wants to fetch some food from the refrigerator. Until enough evidence for either possibility is observed, both plans remain equally valid.

Even sketchy hypotheses contain enough information to generate expectations about possible next actions. If expectations are violated, revision becomes necessary in order to reconcile observed actions with the new hypotheses. In the kitchen example, if the actor leaves the house through the kitchen, the observer must revise her set of plausible hypotheses to include "going for a walk", given the new observation.

Depending on the role of the agent in the process, plan recognition can be further subdivided. In *keyhole recognition* the agent's plan is recognised from passive observations only. The agent does not behave in any way to facilitate or obstruct the recognition. This type of recognition prevails among help systems which generate unsolicited advice. During *intended recognition* the agent helps the system to uncover her plans. This is necessary, for example, in language understanding where the system must recognise the agent's plan in order to generate an appropriate response. Conversely, an agent can also thwart recognition to hide her plans, which might occur in adversary settings such as warfare or gaming situations.

3.1.1 Plan recognition methodology

Plan recognition assumes the agent is pursuing a set of goals. The main task of a plan recognition system is to infer those goals from a set of observed actions. In order to achieve this, the system is provided with a domain-specific plan library, which encodes the actions the agent might execute to reach her goals. The plan library comprises a set of actions, their preconditions, certain constraints, and the consequences of executing each action in the form of goals and secondary sub-goals.

One of the earliest applications of formal reasoning on such a plan library is described in (Kautz and Allen, 1986). These authors propose a theory for plan recognition based on deductive inference. It combines observations, an action taxonomy, and a set of explicit constraints to deduce the agent's plan through logic reasoning. The action taxonomy organises actions into a hierarchy of abstractions and decompositions. Abstraction connects a set of specialised actions with an abstract parent action. For instance, specialised instances of "travelling" can be "travelling by train" and "flying". Furthermore, actions are decomposed into steps. "Travelling by train" consists of the sub-actions, "purchasing ticket", "waiting for the train at the station", and "riding the train to the destination".

The action taxonomy of (Kautz and Allen, 1986) incorporates two *closed-world assumptions*. Firstly, all ways of performing a certain action or plan are known. Secondly, all actions have a purpose and this purpose is known. Besides the closed-world assumptions, an additional axiom is necessary to recognise the agent's plan through logical deduction. This axiom requires a minimal number of top-level self-explanatory plans.

A plan library is linked to a particular domain, but plan inference itself is largely domain-independent. (Carberry, 2001) describes plan inference as the process of chaining actions and sub-goals until an observed action can be connected to one of the possible domain goals. Starting from the currently observed action, the plan recognition system determines the result of executing that action. This result, in turn, is connected to all possible actions having it as a precondition. The process iterates until reaching one of the domain goals. Each path connecting the current action to one of the domain goals offers an explanation of how that action contributes to realising a particular goal (Carberry, 2001).

3.1.2 Restricting hypotheses

The process of action chaining most likely leads to multiple plausible hypotheses, which are all consistent with the observed action. In the literature, a multitude of heuristics have been proposed to narrow the space of hypotheses and most are based on rationality and coherence. However, (Carberry, 2001) claims that the effectiveness of individual heuristics has yet to be evaluated in large-scale real-world applications.

One such heuristic includes domain information to favour certain plan hypotheses over less plausible alternatives. A speaker asking about the nearest post office most likely intends to send a letter. However, alternative a priori less plausible options exist. For instance, the speaker might want to apply for a job at the post office.

(Carberry, 1990) proposes a natural language consultation system based on Dempster-Shafer reasoning. Inspired by psychological studies of human inference during conversations, the system sanctions default interpretations, but updates the estimation of the agent's plan if further evidence favouring other hypotheses is provided. The evidence of different pieces of information is combined in a Dempster-Shafer framework for reasoning under uncertainty.

Another way of narrowing the hypothesis space is to consider group-specific patterns. (Gertner, 1997) proposes an online critique system for assisting surgeons during intensive care operations. The developed critique system analyses the actions of the surgeon in order to infer her plan and produce specific advice. As the doctor is an expert in the domain, her plans might be suboptimal rather than wrong. This heuristic biases the plan recognition towards hypotheses including fewer incorrect goals. Plan recognition is tailored to a particular user group rather than to the individual user of the system, which will be the topic of the next section.

A student-tutoring system has different requirements since novice users are being trained. Errors are frequent and therefore, the tutoring-system should focus also on hypotheses including incorrect goals.

3.1.3 User adaptation

Adapting plan recognition can improve performance by considering the behaviour of a particular user explicitly. In (Bauer, 1994; Bauer, 1996) a plan recognition system is proposed that includes user preferences to evaluate competing hypotheses. User preferences are recorded during typical interactions of a user with her email client. In this domain, repetitive interactions serve as a basis to predict future behaviour from observations. Since a certain action might appear in several plans, Bauer's system relies on Dempster-Shaefer theory to model plan uncertainty. Bauer justifies his choice for the Dempster-Shaefer framework by the fact that uncertainty can be assigned to a set of compatible plans while maintaining ignorance about its specific members. Furthermore, Dempster-Shaefer can handle observations of the user's interaction through an observation mapping. This mapping assigns to each non-empty subset of the observation space a set of plan hypotheses and numerical values indicating the strength of connection between observations and plans. In (Bauer, 1994) heuristics for initialising and updating the observation mapping are provided. The main drawback of the Dempster-Shafer framework is the lack of unique ordering criterion for the calculated uncertainties. Dempster-Shafer theory reasons in terms of uncertainty intervals, which cannot be ordered if they overlap. Therefore, making decisions to aid the user is not straightforward. Furthermore, the exponential complexity of the evidence and hypothesis spaces in

Dempster's update rule make it impractical for real world applications with a high number of actions and plans.

User adaptation has also been explored in the literature without any reference to uncertainty on plan hypotheses. (Lesh, 1997) proposes a goal recognition system, which adapts to the observed behaviour of a user by adding and removing goals from two sets: the background goal set and the potential goal set. Including goals in the potential goal set should increase prediction accuracy since the plan recognition system can choose among more possibilities. However, Lesh argues that a large set of potential goals reduces efficiency if more incorrect hypotheses have to be discarded. The background goal set increases the system's robustness by reducing the influence of noise. Spurious actions introduced in the agent's plan no longer lead to incorrect classifications as long as they are recognized as belonging to background goals.

The plan recogniser of (Lesh, 1997) is adapted to the user via unsupervised greedy gradient ascent. The optimisation compares different sets of background and potential goals in terms of accuracy and coverage. Lesh evaluated the algorithm on simulated data of agents performing a limited set of tasks. In the reported experiments, adapting goal recognition to different users increased performance compared to the generic recognition. However, even if the approach could be transferred to other plan recognition systems, discarding the uncertainty severely limits its applicability to ambiguous real world settings.

3.1.4 Bayesian approaches

The first application of Bayesian plan recognition dates back to the work by (Charniak and Goldman, 1993; Charniak and Goldman, 1991) on natural language story understanding. The plan recogniser, Wimp3, processes written sentences to construct a Bayesian Network that can incorporate probabilistic inference.

Charniak and Goldman identify three major shortcomings of previous work by (Kautz and Allen, 1986), which their system overcomes. Firstly, Kautz and Allen disregard prior probabilities of different plan hypotheses. In Kautz and Allen's system highly unlikely and very probable plans are indistinguishable as long as they explain the observed action sequence. Secondly, Kautz and Allen's system minimises the number of top-level plans. This poses the problem of how to specify a distinction between top- and low-level plans. Thirdly, Charniak and Goldman doubt the validity of minimisation as an abduction criterion. Often, they claim, it is preferable to allow a set of valid hypotheses rather than committing to a single, possibly unlikely, explanation.

In (Charniak and Goldman, 1993), root nodes of the Bayesian Network correspond to hypothesised plans. However, as the story progresses, former root nodes can become itself part of higher-level plans because the Bayesian Network is constructed

iteratively. The problem of plan recognition reduces to constructing and evaluating these Bayesian Networks, where domain actions and logical predicates are modelled as random variables. Wimp3 parses each sentence in a story by introducing new nodes for detected actions and by linking these actions to already existing nodes. Gradually, actions and their describing properties organise into a plan hierarchy, where links between nodes denote the evidence lower nodes provide for parent nodes in the hierarchy.

Network construction is guided additionally by a marker passing algorithm, which avoids uncontrolled growth of the Bayesian Network. For instance, some actions might be explained in a multitude of ways and thus crowd the graph hierarchy with edges. Recursive plans, on the other hand, cannot be translated into an acyclic graph. The proposed marker passing algorithm propagates marks through neighbouring nodes in the Bayesian Network and detects all intersections where marks originating from different nodes meet. Only the paths between endpoints of the algorithm trigger network extension rules in order to bound complexity. These extensions reflect detected relations between observed and hypothesised entities, which can result either in hypothesising new plans or in connecting already existing hypotheses.

When used for plan inference, Wimp's Bayesian Network infers the probability of each proposition via Bayes' rule by combining causal evidence of parent nodes with diagnostic evidence of child nodes. Adding new evidence to the network triggers update rules, which recompute the probability at each node. Charniak and Goldman's system requires a large number of prior and conditional probabilities, which ideally should be obtained from data. However, it is unclear how these probabilities can be estimated reliably in the proposed application of story understanding. Furthermore, Wimp3 is insensitive to the order in which actions are presented to it. Arguably, this can be a limitation or advantage of the system depending on the story's complexity and the narrative structure.

In another application of Bayesian Networks, (Albrecht et al., 1998) estimate the parameters of a network to perform keyhole plan recognition in an adventure game. Their Dynamic Bayesian Network calculates three conditional probability distributions for the set of next actions, for next locations and for players' next goals given previously seen locations and actions. The system learns how the next goal can be explained statistically in terms of actions carried out at certain locations, and how actions and locations follow each other. The challenge of this application is the large action and location sets containing several thousands of items. Therefore, the authors propose four simple Bayesian Networks of increasing descriptive power to perform Bayesian plan recognition. However, they fail to address the problem of identifying the structure of the Bayesian Network automatically, but provide means to compare different network topologies.

(Albrecht et al., 1998) achieve prediction gains by reducing the set of actions. The system classifies an action into spurious or relevant during preprocessing of the data. Further, simplifying the set of locations reduces computational time and

memory requirements at little expense of predictive accuracy. These simplifications include aggregating visited locations and considering all similar locations together regardless of their situation in the environment.

Despite variations in prediction performance across models, an important feature of the approach of (Albrecht et al., 1998) is its robustness against plan recognition failures by spurious actions or by actions performed across many goals. Given the statistical training procedure, only those actions normally leading to complete a goal and, those which are executed by many users, contribute significantly to the predictions.

3.1.5 Robustness

Robustness is a major issue for plan recognition, especially due to the presence of noise in the inputs. Noise appears on the agent's input either by extraneous actions, which do not contribute to the currently pursued goal, or by inappropriate actions resulting from misconceptions about means of achieving a goal. The latter corresponds to semantic confusion, which is more difficult to detect and overcome. Semantic confusion is included as noise source since it corrupts the plan recognition process.

An agent might also deliberately hide her plan or mislead the plan recognition system in adverse situations. Another source of noise affecting plan recognition resides in errors while constructing the set of possible hypotheses. In this case, the new input of the agent might not be integrated correctly into the partially assembled plan.

Furthermore, plan recognition is hampered with incomplete plan libraries if the system cannot incorporate correct novel plans. This problem is very difficult and has received little attention in research (Carberry, 2001).

Probabilistic approaches to plan recognition offer advantages over non-probabilistic approaches when the system's beliefs need to be revised and multiple hypotheses are maintained simultaneously. However, even probabilistic approaches require mechanisms to deal with user misconceptions and novel plans.

3.1.6 Applications

Applications of plan recognition systems have been studied in the fields of language understanding and automatic response generation (Allen and Perrault, 1980), speech-to-speech translation (Alexandersson, 1995), computer-aided design interfaces (Goodman and Litman, 1992), Unix help systems (Mayfield, 1992), collaborative problem solving (Lesh et al., 1999), and automated description of image sequences (Retz-Schmidt, 1991).

3.2 Probabilistic Human-Robot Navigation

As a first step in understanding human-robot navigation, we will develop a probabilistic model for the *navigation assistance* architecture presented in the introduction. For the reader's convenience, we repeat the overall architecture in figure 3.1.

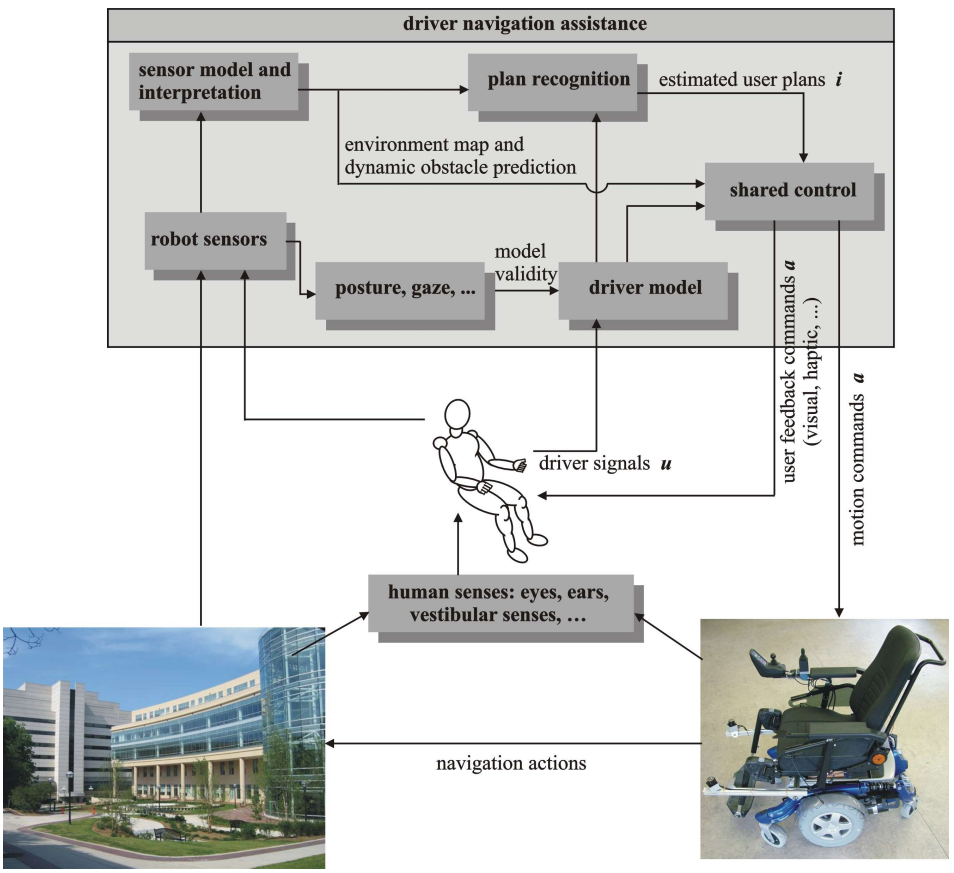


FIGURE 3.1: Framework for driving assistance

The probabilistic view of *driver navigation assistance* includes a model for *sensor perception*, *plan recognition* and the *human driver*. We defer the discussion about *shared control* to chapter 5.

The robot operates in a potentially unknown environment and ignores its location within this environment. Further, it interacts with a human who does not communicate her plans explicitly. Therefore, the robot needs to estimate three

uncertain quantities: the map of the environment, $\boldsymbol{m}$, its pose with respect to the map, $\boldsymbol{x}$ and the user's plan $\boldsymbol{i}$. The robot's pose $\boldsymbol{x}$ as well as user plans $\boldsymbol{i}$ change over time.

We denote the complete sequence of robot poses until time k by $\boldsymbol{x}_{0:k} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k\}$ and similarly, the sequence of user plans by $\boldsymbol{i}_{0:k} = \{\boldsymbol{i}_0, \ldots, \boldsymbol{i}_k\}$. In order to estimate its state, the robot is able to access uncertain projections of this hidden state through an array of sensors. Measurements about the external environment are termed $\boldsymbol{z}$, and user inputs, which are measurements about the user are $\boldsymbol{u}$. The robot then selects actions $\boldsymbol{a}$ which cause transitions in state.

Figure 3.2 shows the complete state estimation problem the robot faces when interacting with its human operator. We model this state estimation as a Dynamic Bayesian Network (DBN). This Dynamic Bayesian Network specifies the probabilistic relations between random variables, which the robot can observe or has to estimate. It also incorporates the flow of time from left to right in the model. A random variable is represented inside a circle or node, which is either shaded for observed variables or uncoloured for hidden variables. The complete Dynamic Bayesian Network represents a factorisation of the joint probability over all variables present in the model. Each individual factor is a conditional probability of a child node conditioned on its parent nodes, as specified by the direction of the edges. Therefore, directed edges indicate probabilistic dependencies between variables. Please refer to chapter 2 for further details on the semantics and computations in Dynamic Bayesian Networks in general. The specific probabilistic relationships entailed by the human-robot navigation DBN are the subject of the remainder of this section.

In joint human-robot navigation the evolution of user plans governs the time scale of the robot's Dynamic Bayesian Network. The robot needs to query its sources of information, i.e. its external sensors $\boldsymbol{z}$, the user sensors $\boldsymbol{u}$, the localisation $\boldsymbol{x}$ and map-building $\boldsymbol{m}$ capabilities at the bandwidth of the user in order to estimate the user's plans. Section 3.2.2 provides additional information about this time scale.

As the DBN of figure 3.2 implies, the robot needs to calculate the joint probability distribution over trajectory $\boldsymbol{x}_{1:k}$, environment map $\boldsymbol{m}$, and evolution of user plans $\boldsymbol{i}_{0:k}$, given the sequence of measurements $\boldsymbol{z}_{0:k}$, user signals $\boldsymbol{u}_{0:k}$, and robot actions $\boldsymbol{a}_{1:k}$.

$$\text{probabilistic human-robot navigation: } p\big(\boldsymbol{m}, \boldsymbol{x}_{1:k}, \boldsymbol{i}_{0:k} \mid \boldsymbol{u}_{0:k}, \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) \quad (3.1)$$

For a practical implementation of the architecture in figure 3.1, it is interesting to consider different parts of the human-robot DBN in figure 3.2 running at different frequencies. For instance, the external sensors of the robot could provide information at a higher frequency in order to construct more accurate maps. This information could then become available to other navigation modules with independence of the user's plans. For instance, an autonomous trajectory planner (Demeester

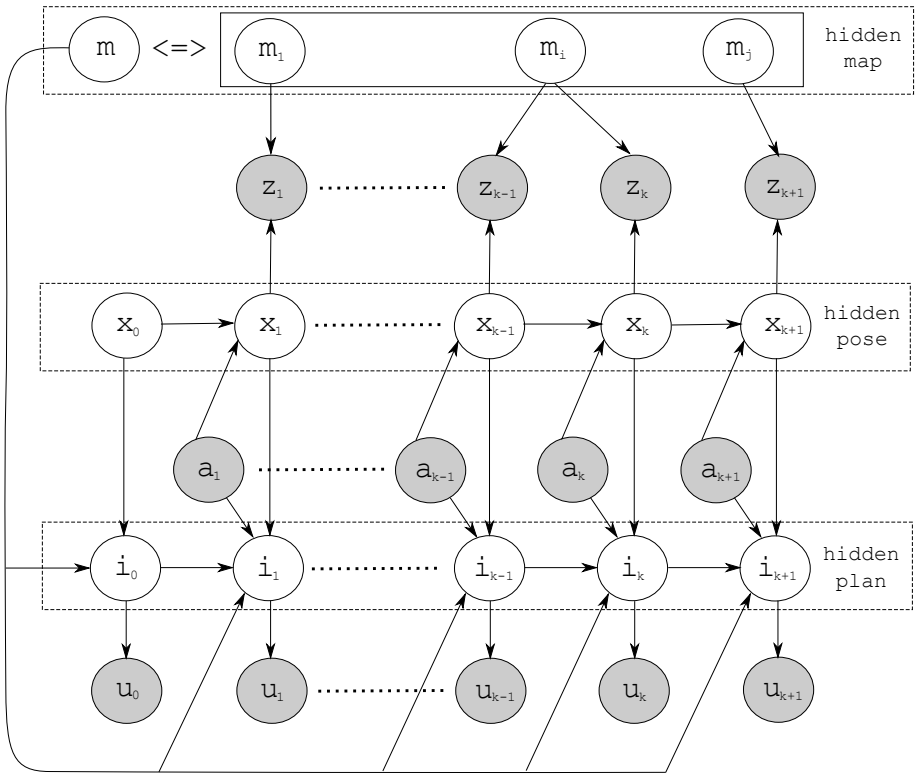


FIGURE 3.2: Dynamic Bayesian Network representing the complete state estimation problem the robot faces. The robot moves in an unknown environment and has to estimate the plan of its user. $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N)$ is the uncertain environment map of features $\boldsymbol{m}_i$, $\boldsymbol{x}$ the robot's pose within this map, $\boldsymbol{i}$ represents the user's plan, $\boldsymbol{z}$ are measurements of external sensors, $\boldsymbol{u}$ user inputs, and $\boldsymbol{a}$ robot actions.

et al., 2005) does not require knowledge of the user's intentions in order to plan trajectories to far-away destinations.

Different time scales in the human-robot navigation DBN effectively splits the complete state estimation problem of equation 3.1 into two marginal probability distributions. The first marginalisation eliminates the user plan evolution of equation 3.1 leaving a joint distribution of map, $\boldsymbol{m}$ and robot positions, $\boldsymbol{x}_{0:k}$. Calculating this joint distribution is called *Simultaneous Localisation And Mapping* (SLAM).

$$\text{SLAM: } p\big(\boldsymbol{m}, \boldsymbol{x}_{1:k} \mid \boldsymbol{u}_{0:k}, \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) \tag{3.2}$$

The second marginalisation deals exclusively with user plans, assuming perfect knowledge of the environment and the robot's localisation. We call this process *plan recognition*, by which the robot tries to estimate the user's plan in order to provided user-specific assistance.

$$\text{plan recognition: } p\big(\boldsymbol{i}_{0:k} \mid \boldsymbol{x}_{0:k}, \boldsymbol{m}, \boldsymbol{u}_{0:k}, \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}\big) \tag{3.3}$$

3.2.1 Simultaneous Localisation and Mapping (SLAM)

We will briefly discuss SLAM in the probabilistic model of this architecture, but we refer the interested reader to the extensive overview of robot perception and SLAM by (Thrun et al., 2005) and to the tutorials (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006).

When a robot moves through an unknown environment taking relative observations of landmarks, these observations become all correlated with each other due to the common error in estimating the robot's location. As a consequence, in order to construct a consistent map, it is necessary to consider the joint state vector of robot and landmark locations. This joint state vector needs to be updated with every landmark observation. The dimensionality of the joint state vector is thus of the order of the number of landmarks in the map, with computation scaling as the square of the number of landmarks. The combined localisation and mapping problem once formulated as a combined estimation problem converges. In fact, the correlations between landmarks are the critical part of the SLAM problem. The more these correlations grow, the better the solution (Durrant-Whyte and Bailey, 2006).

Let us consider now the structure of the SLAM problem as modelled by the Dynamic Bayesian Network of figure 3.2. The *sensor model* describes the probability of observing landmarks $\boldsymbol{z}_k$ from position $\boldsymbol{x}_k$ assuming complete knowledge of the map $\boldsymbol{m}$ and of the robot's current pose $\boldsymbol{x}_k$:

$$\text{sensor-model: } p\big(\boldsymbol{z}_k \mid \boldsymbol{x}_k, \boldsymbol{m}\big) \tag{3.4}$$

Complete knowledge of the map and of the robot's current position makes observation $\boldsymbol{z}_k$ conditionally independent of other prior and future observations. Additional observations cannot improve the knowledge obtained from the map and the current robot pose. In figure 3.2 observation $\boldsymbol{z}_k$ depends only on the map and the current robot position.

The *motion model* summarises transitions of robot poses. If we hypothesise that the robot's current pose depends uniquely on its previous pose and its current action $\boldsymbol{a}_k$, then the motion model becomes:

$$\text{motion model: } p\big(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{a}_k\big) \tag{3.5}$$

Posterior SLAM. We are now in place to propose a sequential solution to the SLAM problem, which computes the joint probability of current robot position and map from past estimates:

SLAM time-update:

$$p^k_{prior}\begin{pmatrix}\boldsymbol{x}_k\\ \boldsymbol{m}\end{pmatrix}\begin{vmatrix}\boldsymbol{z}_{0:k-1}\\ \boldsymbol{a}_{1:k}\\ \boldsymbol{x}_0\end{vmatrix} = \int p\big(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{a}_k\big) \cdot p^{k-1}_{post}\begin{pmatrix}\boldsymbol{x}_k\\ \boldsymbol{m}\end{pmatrix}\begin{vmatrix}\boldsymbol{z}_{0:k-1}\\ \boldsymbol{a}_{1:k-1}\\ \boldsymbol{x}_0\end{vmatrix} d\boldsymbol{x}_{k-1} \tag{3.6a}$$

SLAM measurement-update:

$$p^k_{post}\big(\boldsymbol{x}_k, \boldsymbol{m} \mid \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) = \frac{p\big(\boldsymbol{z}_k \mid \boldsymbol{x}_k, \boldsymbol{m}\big) \cdot p^k_{prior}\big(\boldsymbol{x}_k, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big)}{p_{norm}\big(\boldsymbol{z}_k \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big)} \tag{3.6b}$$

The SLAM time-update in equation 3.6a propagates the SLAM posterior at time $k-1$ to time k by considering the present robot action $\boldsymbol{a}_k$, but without reference to the current measurement $\boldsymbol{z}_k$. The arrangement of variables follows from constraints in the layout, but lacks any special semantics. Equation 3.6a still relates the multivariate probabilities of map $\boldsymbol{m}$ and pose $\boldsymbol{x}_k$ conditioned on robot actions $\boldsymbol{a}_{1:k}$, sensor measurements $\boldsymbol{z}_{0:k-1}$ and the first pose $\boldsymbol{x}_0$.

The *SLAM time-update* results from marginalising out pose $\boldsymbol{x}_{k-1}$ from the distribution $p\big(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big)$ after applying the product rule as

follows:

$$p^{k}_{prior}\big(\boldsymbol{x}_k, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) =$$

$$\overset{\text{marg.}}{=} \int p\big(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) d\boldsymbol{x}_{k-1} \tag{3.7a}$$

$$\overset{\text{prod.}}{=} \int p\big(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{m}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) \cdot p\big(\boldsymbol{x}_{k-1}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0\big) d\boldsymbol{x}_{k-1} \tag{3.7b}$$

$$\overset{\text{indep.}}{=} \int p\big(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{a}_k\big) \cdot p^{k-1}_{post}\big(\boldsymbol{x}_{k-1}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k-1}, \boldsymbol{x}_0\big) d\boldsymbol{x}_{k-1} \tag{3.7c}$$

In equation 3.7a we eliminate variable $\boldsymbol{x}_{k-1}$ from the joint posterior over the robot's current pose, its past pose and the map. The next equation, eq. 3.7b, follows from the product rule. In the last equation, eq. 3.7c, we have applied two conditional independence relations. First, the current robot pose is conditionally independent of the map, the sequence of actions up to time $k-1$ and any sensor data. In fact, as the motion model implies, the robot pose transition is a first order Markov process that depends only on the previous pose and action $\boldsymbol{a}_k$. Further, the posterior at the previous time step, $p^{k-1}_{post}(\boldsymbol{x}_{k-1}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{1:k-1}, \boldsymbol{x}_0)$ is independent of action $\boldsymbol{a}_k$ in the future. These independence relationships are entailed in the Dynamic Bayesian Network of figure 3.2.

The SLAM measurement-update in equation 3.6b follows from Bayes' rule and the conditional independence relationships of the *sensor model*. It describes how sensor observations $\boldsymbol{z}_k$ are introduced in the model and correct the posterior. Successive measurements can be considered "nearly independent" observations of the relative locations between landmarks. Even though the location of specific landmarks might be uncertain, the relative position between them can be known with high certainty. The landmarks' uncertainty originates from the same source, namely estimation errors of the robot's position, but this source of uncertainty is shared among all landmarks (Durrant-Whyte and Bailey, 2006).

Solutions to the SLAM problem depend on the representation of *sensor* and *motion models*. One possible solution is a state-space representation of sensor and motion models with additive white Gaussian noise, leading to the use of Extended Kalman Filters. This solution is known as EKF-SLAM (Dissanayake et al., 2001; Thrun et al., 2005).

An alternative solution represents the motion model by a Rao-Blackwellized particle filter, which can deal with arbitrary non-Gaussian distributions. This

implementation is called *FastSLAM* (Montemerlo et al., 2002; Montemerlo and Thrun, 2003). The reason for its speed, hence the name *FastSLAM*, stems from the factorisation of the joint posterior probability over the complete trajectory and the map:

$$p(\boldsymbol{x}_{0:k}, \boldsymbol{m} \mid \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0) = p(\boldsymbol{m} \mid \boldsymbol{z}_{0:k}, \boldsymbol{x}_{0:k}) \cdot p(\boldsymbol{x}_{0:k} \mid \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_0) \quad (3.8)$$

Conditioning on the complete trajectory renders the different map features $\boldsymbol{m}_i$ conditionally independent as can be checked in the graphical model of figure 3.2. This allows to represent the map by a set of independent Gaussians instead of a joint Gaussian of quadratic complexity.

$$p(\boldsymbol{m} \mid \boldsymbol{z}_{0:k}, \boldsymbol{x}_{0:k}) = \prod_{j}^{M} p(\boldsymbol{m}_j \mid \boldsymbol{z}_{0:k}, \boldsymbol{x}_{0:k}^{(i)}) \quad (3.9)$$

where $\boldsymbol{x}_{0:k}^{(i)}$ are particles in the Rao-Blackwellized particle filter. The disadvantage of the *FastSLAM* algorithm is that particles deplete over time.

3.2.2 Plan recognition and user modelling

The purpose of our probabilistic plan recognition framework is to recognize *all* possible user navigation intents, to estimate the uncertainty on these intents, and to make the estimation adaptive to the specific user that is interacting with the robot. In order to realize this, a representation of user intents is chosen that is different from previous approaches, which nearly all represent intents in terms of specific assistance algorithms such as *follow-corridor* or *avoid-obstacle*.

Representation of user plans. User intentions or user plans $\boldsymbol{i}$ can be generically described as a certain goal twist $\boldsymbol{t_{goal}} = [v_{goal}\ \omega_{goal}]^T$ that the user wishes to achieve at a certain goal pose $\boldsymbol{x}_{goal} = [x_{goal}\ y_{goal}\ \theta_{goal}]^T$, where v_{goal} denotes the desired linear velocity and ω_{goal} the rotational velocity at $\boldsymbol{x}_{goal}$. The trajectory that the user has in mind to achieve this goal pose and goal velocity is also part of the intention representation. It is imperative to consider the trajectory as part of the user plan since a user can reach the same goal pose in a myriad ways.

Any intention can be modelled with this representation in a precise way, to *any* place in the robot's environment close by or far away.

Plan recognition proceeds in two layers. At the lowest level, the robot needs to track continuous user plans $\boldsymbol{i}$ to navigate safely. Section 3.2.3 discusses how the robot generates a continuous plan library $\boldsymbol{\mathcal{I}}$ adapted to obstacles in the environment. In

order to reason about the plans of the user, the robot also needs a deliberative layer, which calculates the probability of a set of hypotheses given past driving behaviour and the present input of the user. This deliberative layer corresponds to the problem of *plan recognition* and will be explained next.

Probabilistic plan recognition. The link between continuous user plans and the deliberative plan recognition layer rests on the assignment of probability mass over the space of user plans $\boldsymbol{i}$. The following equation states how probabilities of user plans are mapped to a continuous multidimensional space of poses.

$$p\big(\boldsymbol{i}, \boldsymbol{x}_k\big) = p\big(\boldsymbol{x}_k, \boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_m^i\big) = p\big(\boldsymbol{x}_k\big) \cdot \prod_j^m p_{gen}\big(\boldsymbol{x}_j^i \mid \boldsymbol{x}_{j-1}^i, \ldots, \boldsymbol{x}_1^i\big) \tag{3.10}$$

where $p_{gen}(\cdot)$ defines the generative model that distributes probability mass and $p\big(\boldsymbol{x}_k\big)$ represents the robot's localisation uncertainty. Other generative models are conceivable, such as probability assignments to the set of robot controls or to the set of continuous parametrised trajectories.

It is possible to consider that users might not build and maintain a very precise numeric model of their intention trajectory. Imprecise user plans distribute probability mass over large volumes. Conversely, precise plans are more concentrated around kinematically and dynamically feasible trajectories.

Equation 3.10 introduces probabilistic semantics in the process of generating user plans. In order to reason about the user's plan, it is still necessary to define a plan library $\boldsymbol{\mathcal{I}}$. Several options are open. The robot could reason in a continuous hypothesis space, where probability distributions of user plans are defined over continuous trajectories. Alternatively, the robot could infer the user's plan out of a discrete set of hypotheses. In order to estimate the intention of the user correctly this set must contain all conceivable user plans within a certain resolution, i.e. the hypotheses need to be *exhaustive*. In this work we follow the latter approach.

Further, we impose that the generative model $p_{gen}(\cdot)$ in equation 3.10 respects the robot's kinematics and dynamics, and only *safe* positions are allowed. Arguably, these are strong assumptions since users might have erroneous ideas of the robot's behaviour. However, we have opted to uncouple generation of a complete plan library from the specific driving style of each user. Misconceptions about the robot's behaviour determine the user's driving style, which is captured in a *user model*. This model represents how a particular user translates her plans into inputs to the robot. Consequently, the user model rules out hypotheses of the complete plan library. We describe the user model and its impact on recognising user plans in the following section.

The assumption that the robot's kinematics and dynamics are respected by the generative model $p_{gen}(\cdot)$ restricts the robot's reasoning about user plans only to trajectories which ensure safe navigation.

Before discussing the probabilistic models of plan recognition, it is necessary to consider the temporal evolution of user plans. The preceding discussion about plan generation and probabilistic semantics does not consider temporal relations yet.

We assume that the user has plan $\boldsymbol{i}_0$ in mind when she starts interacting with the robot. However, she only communicates her intentions through an input device, which produces user input $\boldsymbol{u}_0$. The robot observes $\boldsymbol{u}_0$ and moves to pose $\boldsymbol{x}_1$ after executing action $\boldsymbol{a}_1$. When the robot changes its pose after executing action $\boldsymbol{a}_1$, the user adapts by devising a new plan $\boldsymbol{i}_1$, which results in user interface signal $\boldsymbol{u}_1$, and so forth.

$$\boldsymbol{i}_0 \rightarrow \boldsymbol{u}_0 \rightarrow \boldsymbol{a}_1 \rightarrow \boldsymbol{x}_1 \rightarrow \boldsymbol{i}_1 \rightarrow \cdots \rightarrow \boldsymbol{i}_k \rightarrow \boldsymbol{u}_k \rightarrow \boldsymbol{a}_{k+1} \rightarrow \boldsymbol{x}_{k+1} \rightarrow \boldsymbol{i}_{k+1} \rightarrow \ldots \tag{3.11}$$

The flow of time in equation 3.11 imposes a certain sampling rate. This sampling rate can be constant or vary asynchronously. For instance, if a user drives on a long straight corridor without junctions nor obstacles, user plans and user signals might not change for a very long period. Conversely, cluttered indoor environments might probably require frequent adaptations of user plans and input signals. Since plan recognition should be independent of a specific environment, we have selected a constant sampling rate. This constant sampling rate must be compatible with the information processing bandwidth of the user. Otherwise, plan recognition might converge faster than the user can react and effectively cut the user out of the control loop. For the experiments with a continuous user interface of chapter 4 the sampling period is $\Delta T_{sampling} = 0.3\ s$.

Algorithm 4: Plan Recognition: recognise the driving intention of the user

Data: current user input $\boldsymbol{u}_k$, SLAM posterior $p(\boldsymbol{m}, \boldsymbol{x}_k \mid \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k})$, current robot action $\boldsymbol{a}_k$, external sensor data $\boldsymbol{z}_{0:k}$, past posterior $bel(\boldsymbol{\mathcal{I}}_{k-1})$

1 $\boldsymbol{\mathcal{I}}_k$:= generate plan library$\big(p(\boldsymbol{m}, \boldsymbol{x}_k \mid \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k})\big)$

2 **forall** $\boldsymbol{i}_k$ *in* $\boldsymbol{\mathcal{I}}_k$ **do**

3 $\quad p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k\big)$:= evaluate driver model$\big(\boldsymbol{u}_k, \boldsymbol{i}_k, \boldsymbol{z}_{0:k}\big)$

4 $\quad bel(\boldsymbol{i}_k)$:= compute posterior$\big(bel(\boldsymbol{\mathcal{I}}_{k-1}), p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k\big), \boldsymbol{a}_k\big)$

5 **return** *current plan library* $\boldsymbol{\mathcal{I}}_k$*, posterior over current intentions* $bel(\boldsymbol{\mathcal{I}}_k)$

Algorithm 4 summarises the complete plan recognition procedure. Section 3.2.3 explains the generation of a complete plan library $\boldsymbol{\mathcal{I}}_k$ for a specific environment (line 1). The next paragraph describes the driver model in the context of plan

recognition (line 3). The generic algorithm to compute the posterior of line 4 is introduced at the end of this section (section 3.2.2).

User model. User input signals depend directly on user intentions. Relations to any other variable are always mediated through the user's plan, which the user adapts to her current circumstances. We model this dependency by a probabilistic *user model* that corresponds to the likelihood of observed signal $\boldsymbol{u}_k$ given the user has plan $\boldsymbol{i}_k$ in mind.

$$\text{user model: } p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k\big) \tag{3.12}$$

The *user model* captures as a probability distribution the conversion of user plans into user input signals. This model needs to be adapted to every specific user interface and to every user. It is the personalised component of the *navigation assistance* architecture through which user-specific driving assistance becomes possible. Section 3.3.1 will present examples of user models for discrete interfaces and chapter 4 generalises user modelling to continuous interfaces and arbitrary driving disabilities.

Algorithm 5: Driver model: driving behaviour of a specific user

Data: current user input $\boldsymbol{u}_k$, intention $\boldsymbol{i}_k$, external sensor data $\boldsymbol{z}_{0:k}$
1 $\mathcal{UM}$:= select appropriate driver model$\big(\boldsymbol{i}_k, \boldsymbol{z}_{0:k}\big)$ `(optional step)`
2 $\boldsymbol{\Omega}_{\mathcal{UM}}$:= calculate parameters driver model$\big(\boldsymbol{i}_k\big)$
3 evaluate likelihood user input $\boldsymbol{u}_k$ given plan $\boldsymbol{i}_k$:=$p(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{\Omega}_{\mathcal{UM}})$
4 **return** *likelihood user input* $\boldsymbol{u}_k$ *given plan* $\boldsymbol{i}_k$*:* $p(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{\Omega}_{\mathcal{UM}})$

Algorithm 5 summarises all steps related to the online evaluation of the driver or user model. Section 3.2.4 describes the procedure to calibrate a user model for a discrete input device and section 4.3 proposes calibration methods for continuous user interfaces.

Plan process function. Another important component of the architecture in figure 3.1 is the *plan process function*. It reflects how the user adapts her plans in response to robot actions that change the position of the robot. We assume that the user changes her plan $\boldsymbol{i}_{k-1}$ to plan $\boldsymbol{i}_k$ starting from position $\boldsymbol{x}_k$ after observing the effects of action $\boldsymbol{a}_k$. This plan transition is mirrored by the robot in order to understand how user plans are evolving.

Formally, plan transitions are modelled by a *plan process function*, which consists of two parts, one dealing with the continuous layer and the other dealing with the deliberative layer of plan recognition.

The first component adapts all previous user plans $\boldsymbol{\mathcal{I}}_{k-1}$ to trajectories $\boldsymbol{\mathcal{I}}_k$ starting from position $\boldsymbol{x}_k$. Plan adaptation considers the environment map $\boldsymbol{M}$ in order to achieve safe navigation.

The second component of the *plan process function* initialises the probability of the new set of trajectories at time k. This initialisation relies on the *plan process probability*, which serves to calculate a prior over user plans:

$$\text{plan process probability: } p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big) \tag{3.13}$$

Equation 3.13 makes the dependence of the plan process probability on current pose $\boldsymbol{x}_k$ and the extended map $\boldsymbol{M}$ explicit. This extended map $\boldsymbol{M}$ summarises all available information about a certain environment such as a-priori maps, online constructed maps and the history of sensor readings $\boldsymbol{z}_{0:k}$. In practice the dependency on map $\boldsymbol{M}$ and current pose $\boldsymbol{x}_k$ implies that each user plan $\boldsymbol{i}_k$ is adapted to obstacles in the environment. These adapted user plans determine then all probability distributions in the deliberative layer of plan recognition.

Due to the potentially large representation of the map $\boldsymbol{M}$, the plan process probability is not parametrised over the set of possible maps nor over the set of poses. We hypothesize that the adapted user plans $\boldsymbol{\mathcal{I}}_k$ provide a sufficient abstraction of a specific environment. The probabilistic transition to plan $\boldsymbol{i}_k$ from plan $\boldsymbol{i}_{k-1}$ generalises across similar environments with similar plan libraries $\boldsymbol{\mathcal{I}}_{k-1}$ and $\boldsymbol{\mathcal{I}}_k$. In order for this assumption to hold, the plan libraries at time $k-1$ and time k need to be sufficiently rich to represent the topological relationships of a certain local area.

Further, we assume actions $\boldsymbol{a}$ are marginally independent, i.e. they are independent of any other variable. It can be argued that the robot chooses its actions $\boldsymbol{a}$ according to the plan of the user. Action selection is the topic of chapter 5, which describes a framework for *shared control*. Here we are only concerned about the evolution of user plans as a consequence of observed robot actions. The user reacts to the consequences of the displacement and not to the specific choice of actions. These actions could also have been chosen randomly and the user would react in the same way if the sequence of observed displacements is the same.

It is important to highlight the difference between the *plan process probability* and the *user model*. The *plan process probability* quantifies how information from the past propagates to the present without reference to the user's current input $\boldsymbol{u}_k$. As a consequence, the plan process probability is largely independent of a particular user, but reflects how robot actions influence probability assignments to the plan library. Conversely, the *user model* is user-specific. It is a probabilistic model of how a particular user translates her plans into inputs for the robot. The user model corrects the prior over user plans, which originates from the plan process probability, by incorporating user-specific knowledge.

In figure 3.2, the bottom part of the Dynamic Bayesian Network models the evolution of hidden user plans. In order to reduce the complexity of the estimation problem, we assume that robot poses $\boldsymbol{x}_{0:k}$ and the map $\boldsymbol{M}$ are known with absolute certainty during *plan recognition*. For instance, we could consider the mean of the SLAM distribution as a faithful representation of trajectory and map. The Dynamic Bayesian Network for *plan recognition* is reduced then to the graphical model of figure 3.3.

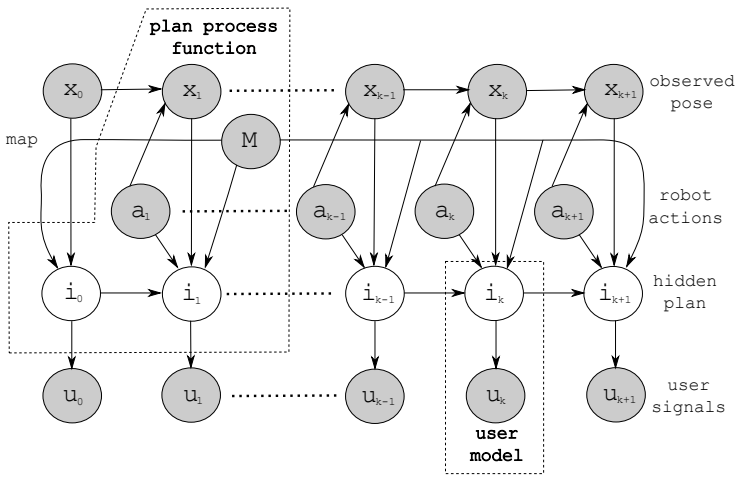


FIGURE 3.3: Dynamic Bayesian Network representing the basic user model and plan process function.

Knowledge about the robot's trajectory and the map are necessary for the trajectory planner that generates the plan library $\boldsymbol{\mathcal{I}}_k$ starting at pose $\boldsymbol{x}_k$. We have opted to assume knowledge about map and pose in order to understand how humans control a robotic wheelchair. Considering localisation and mapping uncertainty introduces additional noise that interferes with analysing human navigation. However, disregarding these uncertainties might lead to unsafe navigation if certain user plans extend into the uncertainty region around environment landmarks or if they start from the wrong pose. Therefore, in order to mitigate the impact of this assumption, it is advisable to include a small safety margin when generating the plan library $\boldsymbol{\mathcal{I}}_k$. In the future, we will extend the framework to a more robust representation of user plans which explicitly deals with localisation and mapping uncertainty.

Posterior plan recognition Similarly to the recursive SLAM solution, we propose a recursive computation of the *plan recognition* probability in equation 3.14.

$$\text{posterior belief: } bel(\boldsymbol{i}_k) \equiv p_{post}\big(\boldsymbol{i}_k \mid \boldsymbol{u}_{0:k}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) \tag{3.14a}$$

plan-recognition time-update:

$$p_{prior}\big(\boldsymbol{i}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) = \sum_{\boldsymbol{i}_{k-1}} p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big) \cdot bel(\boldsymbol{i}_{k-1}) \tag{3.14b}$$

plan-recognition measurement-update:

$$bel(\boldsymbol{i}_k) = \frac{p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k\big) \cdot p_{prior}\big(\boldsymbol{i}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big)}{p_{norm}\big(\boldsymbol{u}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big)} \tag{3.14c}$$

In the preceding equations the map $\boldsymbol{M}$ summarises all available information about the environment, such as a-priori maps, online-constructed maps and the history of external sensor readings $\boldsymbol{z}_{0:k}$, i.e. $\boldsymbol{M} = \{\boldsymbol{m}_{prior}, \boldsymbol{m}, \boldsymbol{z}_{0:k}\}$.

The plan recognition time-update in equation 3.14b propagates the posterior over user plans at time $k-1$ to time k by considering the *plan-process probability*. Again, we can derive the time-update equation by marginalising out variable $\boldsymbol{i}_{k-1}$ from the joint probability of current and prior user plans, which does not yet consider current user signal $\boldsymbol{u}_k$.

$$p_{prior}\big(\boldsymbol{i}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) =$$

$$\overset{marg.}{=} \sum_{\boldsymbol{i}_{k-1}} p\big(\boldsymbol{i}_k, \boldsymbol{i}_{k-1} \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) \tag{3.15a}$$

$$\overset{prod.}{=} \sum_{\boldsymbol{i}_{k-1}} p\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) \cdot p\big(\boldsymbol{i}_{k-1} \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big) \tag{3.15b}$$

$$\overset{indep.}{=} \sum_{\boldsymbol{i}_{k-1}} p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big) \cdot bel\big(\boldsymbol{i}_{k-1}\big) \tag{3.15c}$$

The first step in deriving the prior over user plans at time k is to marginalise out variable $\boldsymbol{i}_{k-1}$ from the joint probability over current and prior plans (equation 3.15a). The joint probability can be factored according to the product rule, as stated in equation 3.15b.

Then, we apply two conditional independence relationships in equation 3.15c. Firstly, knowledge of prior plan $\boldsymbol{i}_{k-1}$, current pose $\boldsymbol{x}_k$, the map $\boldsymbol{M}$ and current robot action $\boldsymbol{a}_k$ renders current plan $\boldsymbol{i}_k$ conditionally independent from prior robot actions $\boldsymbol{a}_{1:k-1}$, prior user signals $\boldsymbol{u}_{0:k-1}$ or prior poses $\boldsymbol{x}_{0:k-1}$. This conditional independence follows from the *plan process probability*.

$$\text{plan process independence: } \boldsymbol{i}_k \perp\!\!\!\perp \{\boldsymbol{a}_{1:k-1}, \boldsymbol{u}_{0:k-1}, \boldsymbol{x}_{0:k-1}\} \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M} \tag{3.16}$$

The second conditional independence involves prior plan $\boldsymbol{i}_{k-1}$ and current robot action $\boldsymbol{a}_k$, which are conditionally independent given prior trajectory $\boldsymbol{x}_{0:k-1}$ and the map $\boldsymbol{M}$. Further, the prior trajectory and the map also d-separate the current pose $\boldsymbol{x}_k$ from the user's prior plan $\boldsymbol{i}_{k-1}$. This conditional independence implies that past intention $\boldsymbol{i}_{k-1}$ is conditionally independent of variables in the future as long as the user's plan in the future $\boldsymbol{i}_k$ and the user's future input $\boldsymbol{u}_k$ remain unknown.

$$\text{past independent of future: } \boldsymbol{i}_{k-1} \perp\!\!\!\perp \{\boldsymbol{a}_k, \boldsymbol{x}_k\} \mid \boldsymbol{x}_{0:k-1}, \boldsymbol{M} \tag{3.17}$$

It can be verified in the Dynamic Bayesian Network of figure 3.14 that no unblocked connection exists between $\boldsymbol{i}_{k-1}$ and $\boldsymbol{x}_k$ or $\boldsymbol{a}_k$ respectively, given prior trajectory $\boldsymbol{x}_{0:k-1}$ and the map $\boldsymbol{M}$. Conditioning on the full prior trajectory $\boldsymbol{x}_{0:k-1}$ maintains this independence relationship also in the complete human-robot navigation DBN of figure 3.2.

The plan recognition measurement-update in equation 3.14c, states how the prior about the user's current plan is corrected by observations, which in this case are measurements acquired through the user's interface, $\boldsymbol{u}_k$. Equation 3.14c follows from Bayes' rule and the conditional independence assumptions about the *user model*.

Posterior plan recognition with extended user model. The experimental results of plan recognition in section 3.3 involve an extended *user model* compared to the one presented previously. This extended *user model* incorporates a longer history of user plans to calculate the likelihood of the user's current input $\boldsymbol{u}_k$. Such model arises if the user bases her input to the robot both on her prior as well as on her current plans. Consequently, the extended user model introduces stabilising feedback in the control of the robot.

Probably, the user's feedback horizon is short because it requires a large memory to store an extensive history of past situations. Further, long planning horizons reduce the capacity of reacting to immediate events. This feedback horizon should not be confused with the implicit knowledge the user accumulates about the behaviour of the robot. Such knowledge could summarise the driving experience of years, but yet depend on a limited window of time when employed to control the robot.

Due to the interest in stabilising plan recognition through temporal feedback, let us assume that user inputs depend on the current and the prior plans, $\boldsymbol{i}_{k-1:k}$. The extend user model is then:

$$\text{extended user model: } p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{i}_{k-1}\big) \tag{3.18}$$

We represent the Dynamic Bayesian Network for *plan recognition* for the extended user model of equation 3.18 in figure 3.4.

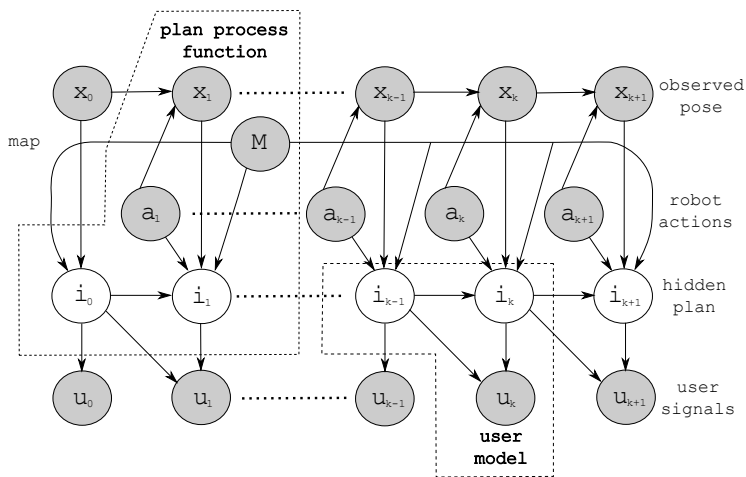


FIGURE 3.4: Dynamic Bayesian Network representing an extension of the basic user model and plan process function.

The extended *user model* of equation 3.18 prevents a formulation of the recursive solution of the *plan recognition* problem in separate prediction and correction steps. Instead, it becomes necessary to combine the time-update and the measurement-update into a single operation because the user model depends also on the user's prior plan.

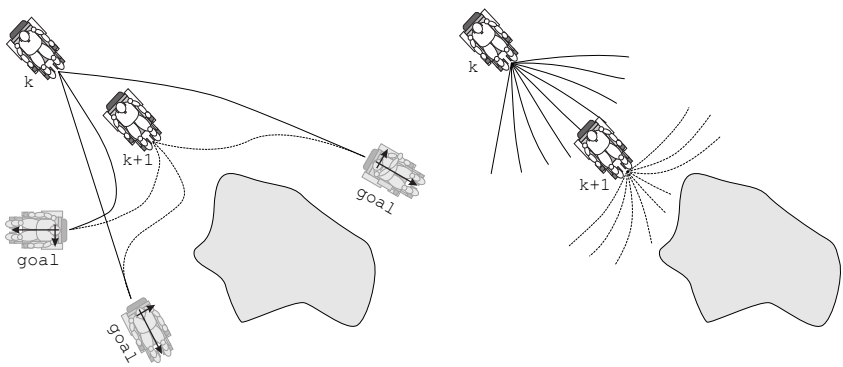
$$bel(\boldsymbol{i}_k) = \sum_{\boldsymbol{i}_{k-1}} \frac{p_{user}\big(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{i}_{k-1}\big) \cdot p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big)}{p_{norm}\big(\boldsymbol{u}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big)} \cdot bel(\boldsymbol{i}_{k-1}) \tag{3.19}$$

where $bel(\boldsymbol{i}_k)$ is the posterior distribution over user plans after considering all observable information at time k and $p_{norm}\big(\boldsymbol{u}_k \mid \boldsymbol{u}_{0:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{x}_{0:k}, \boldsymbol{M}\big)$ is a normalising constant.

The deduction of this equation proceeds along the same lines as for the basic user model. In the extended model of figure 3.4 prior plan $\boldsymbol{i}_{k-1}$ is still conditionally independent of current pose $\boldsymbol{x}_k$ and current robot action $\boldsymbol{a}_k$, given the robot's prior trajectory $\boldsymbol{x}_{0:k-1}$ and the map $\boldsymbol{M}$. This conditional independence holds as long as neither $\boldsymbol{i}_k$ nor $\boldsymbol{u}_k$ are observed.

3.2.3 Generation of user plans

There are basically two strategies for generating a discrete set of plan hypotheses or plan library $\boldsymbol{\mathcal{I}}$, a *global* or a *local* approach. Figure 3.5 compares both strategies and the subsequent paragraphs describe the implications of each approach in detail.



(a) global paths (b) local paths

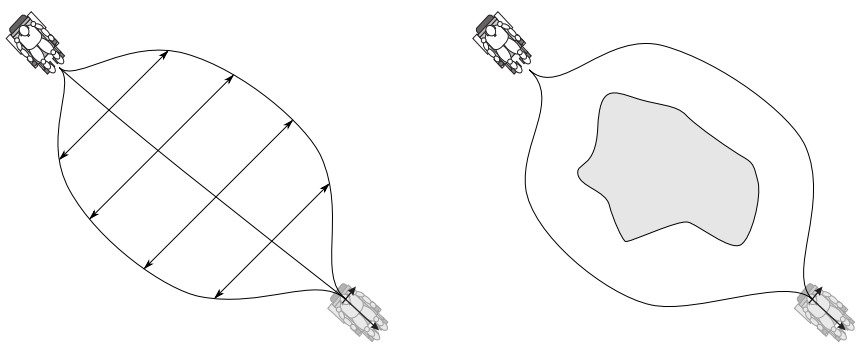
FIGURE 3.5: Global and local approach to generate plan libraries at subsequent times

The distinction between a local plan library and a global plan library becomes more difficult for small environments. If all global destinations are within a small area and these destinations are sufficiently dense, then the global and the local plan libraries at a specific time instant might be virtually indistinguishable. Therefore, it is necessary to define when plan recognition is local or global. The type of plan recognition depends on two properties:

1. Plan generation to global or local destinations. Global destinations are fixed, local destinations change.
2. Temporal reasoning during plan recognition, which defines how information propagates from the past to the present.

Global approach. The robot computes trajectories to a series of goal poses, which might be of interest to the user. These goals can be located at *any* place in the environment, such as at doors, tables, or light switches. Various docking positions to arbitrary obstacles can also be defined. A specific navigation algorithm for corridor-following or docking is no longer required, since driving to any location is possible within the framework.

Generation of goal locations can be either automatic by monitoring places the user visits frequently at a specific time of the day. Goals can also be introduced manually in an a priori map of the environment. In both cases, the total number of goal locations is always finite. Therefore, a necessary constraint to obtain an exhaustive collection of plans, is that *all* homotopically different trajectories the user might follow are plausible hypotheses. Two paths are *homotopic* if a continuous function exists that can transform one path into another while leaving the endpoints fixed (LaValle, 2006).



(a) homotopically equal paths (b) no homotopy

FIGURE 3.6: Transformation of one path into another.

In figure 3.6(a) a continuous transformation of one path into another is possible. Therefore the paths are homotopic. Conversely, there is no continuous function which can transform the paths of figure 3.6(b) while preserving the endpoints fixed. These are two fundamentally different ways of avoiding the obstacle and both ways need to be included as user plans. Homotopic paths, on the other hand, represent local uncertainty, such as the user's imprecise idea of her plans. In the global approach to plan generation, the probability of homotopic paths within a small region are summarised into one single user plan.

The global *plan process probability*, $p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big)$, is straightforward. As figure 3.5 suggests, the prior in the next time step receives its probability from the homotopically equivalent path leading previously to the same goal location. Global goals function as reference points during the time-update step of plan recognition. Thus, the plan process probability is an identity matrix linking equivalent plans

at time $k-1$ to plans at time k. We adopted this technique in (Demeester et al., 2006; Huntemann et al., 2007a; Demeester et al., 2008).

Local approach. The robot computes a different user plan library $\boldsymbol{\mathcal{I}}$. Instead of planning paths to global destinations, a set of local plans is created by integrating various dynamically feasible motor commands, $\boldsymbol{a}^{1:q} = [(v_1, \omega_1), \ldots, (v_q, \omega_q)]$, over a certain amount of time, $\Delta\boldsymbol{T} = [\Delta t_1, \ldots, \Delta t_q]$. This yields a sequence of circular or straight lines starting at the robot's current pose. Across time the destinations of individual plans change. Each plan needs to ensure only the robot's safety, but must not reach a certain fixed goal location. Figure 3.5(b) shows a typical set of local paths.

The advantage of the *local approach* over its *global* counterpart is that navigation assistance is directed to where it is most necessary, namely for local fine-manoeuvring. Local hypotheses can be more dense locally and more imprecise on a global scale, as the user first needs immediate assistance locally. Additionally, a precise global map of the environment is not required as global planning is unnecessary.

The flip-side to *local* hypothesis generation is that temporal reasoning in *plan recognition* becomes more complex. Global references that allow a straightforward initialisation of the prior over user plans are lost. The next section proposes a machine learning technique to derive a local plan process probability $p_{ppf}(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M})$ from observed driving data. This learning approach, which permits plan recognition with local trajectories, is one of the major contributions of this chapter.

Next we consider how the set of local paths is generated online. In order to take moving and unmodelled obstacles into account, the latest laser scan data are incorporated when checking for collisions. Also the robot's location in an estimated map of the environment is taken into account (provided the robot is well localized), by considering the line segments that make up the map. Efficient collision checking is possible for a large number of paths by employing pre-computed look-up tables[1]. Map features and perceived obstacles are placed inside this pre-computed grid when updating the set of local paths. Figure 3.7 shows an example of local plans, adapted to dynamic obstacles at the robot's current location.

The wheelchair's location is estimated as follows. First, a map of the environment is built offline using a feature-based simultaneous localization and mapping (SLAM) algorithm based on single-level relaxation (Frese et al., 2005). Then, for real-time experiments, local localization in the estimated map is performed using an improved version of the Cox scan matching algorithm presented in (Cox, 1989). This algorithm matches laser scans in a least-squares way to segment features in the map. Our modifications are similar to those of (Gutmann et al., 1998),

[1] All collision-free trajectories of a user plan template with 144 plans are calculated in about $20 - 30ms$ on an Intel Core Duo Processor @$2.6GHz$ and $4GB$ of RAM.

i.e. we extract line segments from the scan and use these lines to avoid incorrect assignments of scan points to a priori map lines. The implementation of this localisation algorithm is due to (Demeester, 2007).

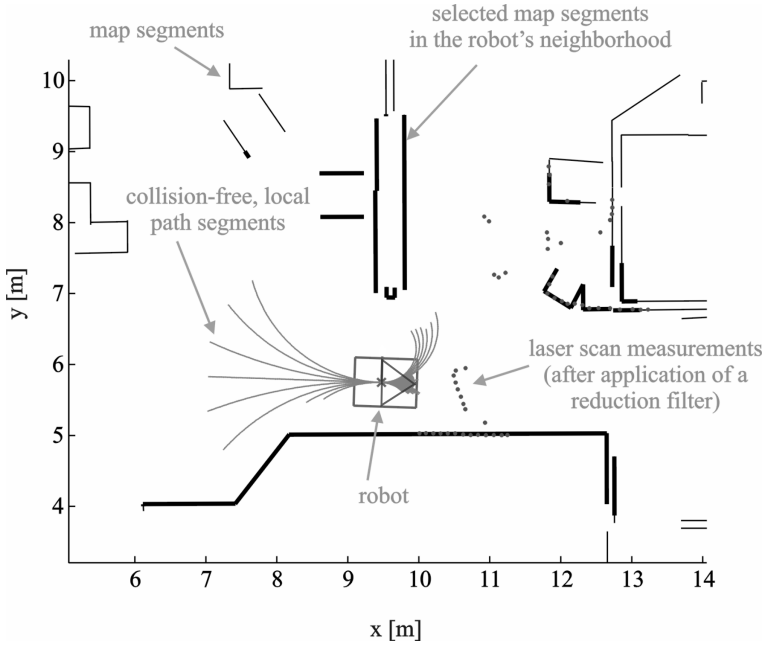


FIGURE 3.7: The figure depicts the wheelchair along with calculated paths to possible goal states at a certain moment in time. Right in front of the wheelchair, an unmodelled object is detected by the laser scanner and taken into account when performing collision checking. The selected map segments that are used for collision checking are shown in bold.

3.2.4 Local plan recognition

The *plan process probability* $p_{ppf}(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M})$ for user plans generated locally does not follow immediately from the destination of user plans at time k and time $k-1$, as in the case of global plans. Indeed, global reference points are lost in the local approach. Therefore, the robot must accept additional uncertainty about the user's intention as a price for the benefit of local driving assistance. The robot needs an alternative strategy to compute a plan process probability, which still allows it to reason about the temporal evolution of user plans.

The *plan process probability* links past user plans $\boldsymbol{i}_{k-1}$ to current user plans $\boldsymbol{i}_k$ and ranges between two extremes. On one end of the spectrum, the plan process probability is completely uniform. In this case, the *plan recognition time-update* (equation 3.14b) which propagates the posterior over plans $\boldsymbol{i}_{k-1}$ to a prior at time k, creates a uniform distribution (cf. equation 3.20).

$$p_{prior}(\boldsymbol{i}_k) = \sum_{\boldsymbol{i}_{k-1}} p_{ppf}(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}) \cdot bel(\boldsymbol{i}_{k-1}) = \kappa \cdot \sum_{\boldsymbol{i}_{k-1}} bel(\boldsymbol{i}_{k-1}) = const \tag{3.20}$$

The posterior coincides then with the user model and any prior information is discarded. Such posterior is more reactive to immediate changes, but the effect of time-filtering is lost and *plan recognition* results in a noisier distribution over user plans.

At the other end of the spectrum, the *plan process probability* is an identity matrix, which links user plans across time with absolute certainty. A user plan at time $k-1$, $\boldsymbol{i}_{k-1}$, corresponds then to a single plan at time k. Absolute certainty is only possible through global knowledge about the final destination of a plan. It could be argued that completely certain assignments of plans are also possible through an identity matrix, which considers the best heuristic correspondences. However, such an approach is brittle and can lead to wrong conclusions in recognising the user's plan. Any identity *plan process probability* in absence of global knowledge leads to an overconfident posterior, which places excessive weight on the correctness of the heuristic that generated the identity matrix.

The key insight that only global knowledge allows absolute certainty, inspires a way to acquire a local *plan process probability*. Figure 3.8 shows the essence of this approach. While driving through the environment, the *robot's path is its destination and nothing more*, paraphrasing a poem by Antonio Machado (Machado, 1912)[2].

A posteriori, during calibration of the plan process probability, the complete trajectory of the robot represents a source of global knowledge. At each sampling step k, different user plans of the plan library $\boldsymbol{\mathcal{I}}_k$ resemble the future path of the robot $\boldsymbol{x}_{k:end}$ to varying extents. Actually, each user plan $\boldsymbol{i}_k$ is a potential future trajectory and can therefore be compared to the actual future trajectory of the robot with the benefit of hindsight.

The comparison between user plans and the robot's future path is independent of a particular user, or of where and how a particular user really intended to navigate. It is merely a comparison between potential and actual future navigation outcomes, i.e. between user plans $\boldsymbol{\mathcal{I}}_k$ and the truly executed future robot trajectory $\boldsymbol{x}_{k:end}$.

[2]The original poem in Spanish and a translation into English can be found in the conclusions chapter, ch. 6.

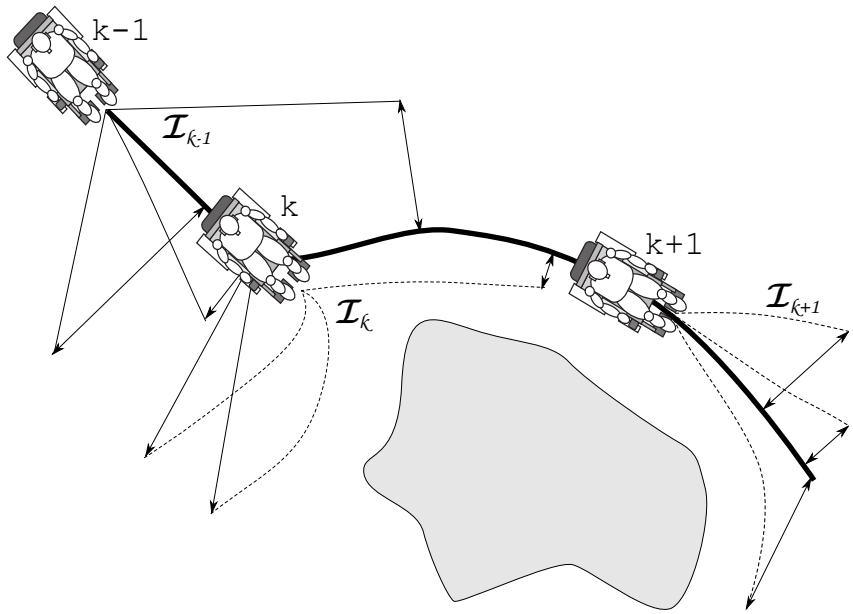


FIGURE 3.8: Plan process probability for local trajectories calibrated with the robot's future path. The bold trajectory represents the path of the robot. At each time step, a set of local trajectories $\mathcal{I}$ emanates from the robot. Arrows between the local trajectories and the robot's future path show a series of soft constraints that define the solution to the local plan process probability.

Section 3.2.2 clarifies the distinction between the user-specific *user-model* and the generic *plan-process probability* further.

In order to calculate the plan process probability, it seems reasonable to assume that equivalent user plans in terms of future path of the robot should inherit probability from their ancestors from previous times. The remainder of the section formalises this intuition into a set of soft probabilistic constraints, which define a solution to the local *plan process probability.*

Trajectory metrics As a first step in computing a local plan process probability, it is necessary to define formally a notion of resemblance between trajectories. This definition should respect metric properties in order to obtain meaningful results, such as that a trajectory is closest to itself or that two very different trajectories cannot be simultaneously close to a third one.

A metric is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following requirements:

- positive: $d(x, y) \geq 0, d(x, y) = 0 \Leftrightarrow x = y$
- symmetric: $d(x, y) = d(y, x)$
- triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$

Computational geometry provides two candidate metrics for comparing user plans, the Hausdorff metric and the Fréchet distance.

Hausdorff distance. If $\mathcal{P}$ and $\mathcal{Q}$ are curves, then the Hausdorff distance is defined as:

$$d_{Hausdorff}(\mathcal{P}, \mathcal{Q}) = \max\{\vec{\delta}(\mathcal{P}, \mathcal{Q}), \vec{\delta}(\mathcal{Q}, \mathcal{P})\} \tag{3.21a}$$

$$\vec{\delta}(\mathcal{P}, \mathcal{Q}) = \sup_{\boldsymbol{x} \in \mathcal{P}} \inf_{\boldsymbol{y} \in \mathcal{Q}} d(\boldsymbol{x}, \boldsymbol{y}) \tag{3.21b}$$

where $\vec{\delta}(\mathcal{P}, \mathcal{Q})$ is the directed Hausdorff distance and $d(\boldsymbol{x}, \boldsymbol{y})$ is a secondary point-metric. The Hausdorff metric measures the maximal displacement, which is possible when travelling on the shortest path from one trajectory to the other. Figure 3.9 shows how to calculate the Hausdorff distance between two trajectories.

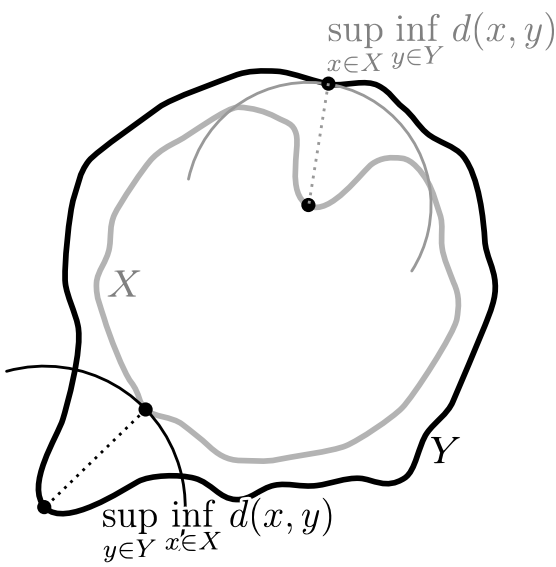


FIGURE 3.9: Calculation of the Hausdorff distance between two curves (Wiki Commons: C. Rocchini, 2011).

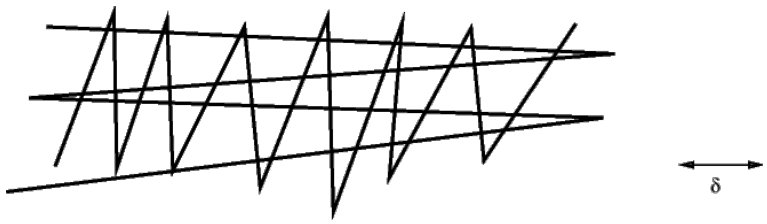


FIGURE 3.10: Comparison of Fréchet and Hausdorff distances (Alt and Godau, 1995). The curves have a small Hausdorff distance, but their Fréchet distance is large.

The arrows in figure 3.8 identify the points farthest away from the robot's future path and whose distance determine the value of the Hausdorff metric.

Besides the brute-force approach of comparing all points pair-wise, an algorithm exists to compute the metric in linear time for any L_p point distance (Shonkwiler, 1991). Efficient algorithms are especially important in image processing when the number of pixels is large. For instance, (Huttenlocher et al., 1993) propose an efficient algorithm to compare images, which calculates the Hausdorff distance between all relative positions of a binary image and a model.

The main drawback of the Hausdorff distance to compare curves is that it neglects the curve's trace. Figure 3.10 shows a pathological example proposed by (Alt and Godau, 1995) of two curves with small Hausdorff distance, but which look rather dissimilar.

The Fréchet distance. An alternative to the Hausdorff metric is the Fréchet metric, which considers the curves' traces when calculating their distance.

$$d_{Fréchet}(\mathcal{P}, \mathcal{Q}) = \inf_{\rho,\sigma} \max_{t\in[0,1]} d\big(\mathcal{P}(\rho(t)), \mathcal{Q}(\sigma(t))\big) \tag{3.22}$$

where $\rho(t)$ and $\sigma(t)$ are continuous and increasing re-parametrisations of the original curves $\mathcal{P}$ and $\mathcal{Q}$. The Fréchet metric calculates the maximal distance between points on the curves for the minimal re-parametrisation. A popular metaphor describes this metric as the minimal length of a leash to walk a dog, when the owner walks on one curve and the dog on the other and neither are allowed to back-trace.

Unfortunately, computing the Fréchet metric is much more involved. Please refer to (Alt and Godau, 1995) for a practical algorithm. However, (Alt et al., 2003) show that for close convex curves both measure coincide. Furthermore, for κ-straight curves whose arc-length between any two points is at most κ times their metric distance, Hausdorff and Fréchet distance differ by a constant factor. In such cases,

the Hausdorff metric provides a reasonable measure of the resemblance between curves, which considers the curves' traces.

In human-robot navigation we assume a rational driver, who might nevertheless drive imperfectly due to her impairment. However, situations such as those in figure 3.10 are completely ruled out from the plan library. Therefore, we will only consider the Hausdorff metric as a measure of resemblance between user plans and the robot's future trajectory.

Pose metrics. The Hausdorff and Fréchet metrics are hierarchical in the sense that they rely on a secondary point-metric. It remains to define an appropriate metric to compare points of a user plan and the robot's future trajectory. Normally, curves are compared with the Hausdorff or Fréchet metrics under a point-particle motion model. Then, the Euclidean distance is a proper measure of metric displacement between curves. However, this assumption is unreasonable for a robot that is capable of turning on-the-spot. Its point displacement can be zero whereas it has turned. The local plan process function should also consider such type of motions.

In order to overcome the limitation of the Euclidean distance for comparing robot motion, we will consider metrics that measure rigid body motion. Comparing rigid body poses is not straightforward, since it is impossible to define a smooth bi-invariant metric on the Euclidean group of rigid body motions $SE(3)$ (Park, 1995). Bi-invariance means invariance both under left and right translations. A left-invariant metric on $SE(3)$ implies:

$$\text{left-invariance: } F_1, F_2 \in SE(3) : d(F_1, F_2) = d(LF_1, LF_2), \forall L \in SE(3) \quad (3.23)$$

Left-invariance reflects the independence of the metric upon the choice of inertial reference frame. Conversely, right-invariance indicates independence of the location of the body-frame attached to the rigid body. It is defined as:

$$\text{right-invariance: } F_1, F_2 \in SE(3) : d(F_1, F_2) = d(F_1R, F_2R), \forall R \in SE(3) \quad (3.24)$$

Finally, as a metric on $SE(3)$ compares positions and orientation, it is desirable that such metric were scale invariant, up to a constant scaling factor. However, no natural length-scale exist for physical space, and, therefore, any metric on $SE(3)$ is parametrised by the choice of length scale (Park, 1995)

Ill-defined metrics depend on the inertial reference frame in which the robot's motion is described. For such metrics, length or angle differ in function of the chosen reference, which is undesirable. (Martinez and Duffy, 1995) propose two frame-invariant metrics to compare rigid body motion, which we discuss next.

Let ν_B be the set of poses of rigid body B, and B_1 and B_2 be two poses, then the first metric by (Martinez and Duffy, 1995) to measure the distance from B_1 to B_2 is:

$$d : \nu_B \times \nu_B \to \mathbb{R} : d_1(B_1, B_2) = \|\delta_M\| = \max_{Q \in B} \|\delta_Q\| = \max_{Q \in B} \|\boldsymbol{r}_{Q1} - \boldsymbol{r}_{Q2}\| \quad (3.25)$$

where $\|\cdot\|$ is the standard Euclidean norm and δ_Q is the displacement vector of an arbitrary point of the finite rigid body B, which at configuration B_1 is located at $\boldsymbol{r}_{Q1}$ and at configuration B_2 at $\boldsymbol{r}_{Q2}$. δ_M represents the displacement of the point farthest away from the screw axis defining the motion. This point is not necessarily unique, but its norm is. In case the motion is a pure translation, all displacements are equal and any point can be chosen to calculate the metric.

The second metric by (Martinez and Duffy, 1995) defines a metric space for the set of planar poses of a finite rigid body.

$$d : \nu_B \times \nu_B \to \mathbb{R} : d_2(B_1, B_2) = \|\delta_M\| + |\lambda \Delta\theta_{1,2}| \quad (3.26)$$

where δ_M coincides with the previous metric, and $\Delta\theta_{1,2}$ is the minimum absolute rotation angle which rotates B_1 into a position parallel to B_2. The length scale λ is defined as the largest distance between any two points in the projection of B onto the displacement plane.

We will employ these metrics as the pose-distance to compare trajectories by means of the Hausdorff metric defined previously. The remainder of this section shows how we convert the distance between the robot's future path and every user plan into a series of soft constraints employed to learn a plan process probability.

Other choices of point distances are possible depending on the parametrisation of user plans. We refer the interested reader to the theoretical study of (Lin and Burdick, 2000) about frame-invariant and objective metrics. They prove that frame-invariance is a sufficient and necessary condition for objectivity of a metric. A metric is objective if it is indifferent to the observer performing metric measurements. Thus, a frame-invariant metric is well defined. (Lin and Burdick, 2000) propose also a series of objective velocity metrics.

Future work will examine how to extend the framework for plan recognition to robots performing arbitrary rigid body motion in space. Then, an alternative representation of user trajectories and metrics to compare motion are needed, which consider, for instance, motion profiles. A possible alternative to describe rigid body motion is the invariant parametrisation of trajectories by (De Schutter, 2010).

Statistical properties of the resemblance between trajectories. The previous section formalises the notion of resemblance between trajectories. Now we need to translate this notion into a procedure to calibrate the local *plan process probability*, $p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \mathbf{M}\big)$.

As a first step, it is interesting to consider the statistical properties of the trajectory metric, which compares plan libraries $\boldsymbol{\mathcal{I}}$ and the robot's future path. According to the discussion in the previous section, we are interested in a probabilistic view of the Hausdorff distance, when individual poses are compared by the metric of (Martinez and Duffy, 1995). The next set of equations summarises the metrics we chose to compare robot trajectories.

$$d_{Hausdorff}(\boldsymbol{i}_k, \boldsymbol{x}_{k:end}) = \max \left\{ \begin{array}{lll} \sup_{\boldsymbol{x}\in\boldsymbol{x}_{k:end}} & \min_{\boldsymbol{y}\in\boldsymbol{i}_k} & d_{MD}(\boldsymbol{x},\boldsymbol{y}) \\ \sup_{\boldsymbol{y}\in\boldsymbol{i}_k} & \min_{\boldsymbol{x}\in\boldsymbol{x}_{k:end}} & d_{MD}(\boldsymbol{x},\boldsymbol{y}) \end{array} \right\} \tag{3.27a}$$

$$d_{MD}(\boldsymbol{x},\boldsymbol{y}) = \max_{B\in robot} \{\|\boldsymbol{x}_B - \boldsymbol{y}_B\|\} \tag{3.27b}$$

$$\|\boldsymbol{x}_B - \boldsymbol{y}_B\| = \sqrt{\left(x_1^B - y_1^B\right)^2 + \left(x_2^B - y_2^B\right)^2} \tag{3.27c}$$

where $\|\cdot\|$ is the standard Euclidean norm, which compares equivalent positions inside the robot's geometry.

In the hierarchy of distances in equation 3.27 the lowest layer, i.e. the Euclidean norm in equation 3.27c, defines the fundamental statistical properties of the complete trajectory metric. The upper layers modify this basic distribution in several optimisation steps.

In the second layer (equation 3.27b), the metric of (Martinez and Duffy, 1995) depends on the motion of the robot to move from a specific pose in user plan $\boldsymbol{i}_k$ to another specific pose on the robot's future trajectory $\boldsymbol{x}_{k:end}$. The Hausdorff metric (equation 3.27a) optimises over all possible poses of the user plan $\boldsymbol{i}_k$ and the robot's future trajectory. The minimisation in equation 3.27a just ensures that the distance between trajectories is unique. However, the relevant optimisation for the Hausdorff metric is a maximisation, which selects the points farthest away from the compared trajectories.

The combined effect of all maximisations is to skew the distribution of the Euclidean norm towards higher distance values, which still maintain large probability. If $d_i \sim p_d(d)$, i.e. the samples of d_i are distributed according to the density $p_d(d)$, then the variable $z = \max\{d_i, i = 1, \ldots, N\}$ maximising over N samples has the following density:

$$z = \max\{d_i, i = 1, \ldots, N\} \Rightarrow p(z) \propto p_d(z) \cdot \left(\int_{-\infty}^{z} p_d(\alpha) d\alpha \right)^{N-1} \tag{3.28}$$

where $\int_{-\infty}^{z} p_d(\alpha)d\alpha$ is the cumulative probability of z under the probability density $p_d(d)$. Equation 3.28 implies that the density $p_d(d)$ gets skewed according to the probability that all $N-1$ samples in the maximisation have a lower value than the maximum of the set, which we denote as z.

According to equation 3.28, it is imperative to understand the properties of the Euclidean norm distribution, since this distribution determines the statistics of the proposed trajectory metric.

A basic result in probability theory establishes that if the coordinates of a vector are independent and identically distributed (i.i.d.) Gaussian random variables of zero mean, the Euclidean norm of this vector has a Rayleigh distribution (Papoulis, 1991).

$$\text{(Papoulis, 1991)} : \boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \sim \mathcal{N}\left[\mathbf{0}; \begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix}\right] \Rightarrow \|a\| \sim \textit{Rayleigh}\,[\sigma] \tag{3.29a}$$

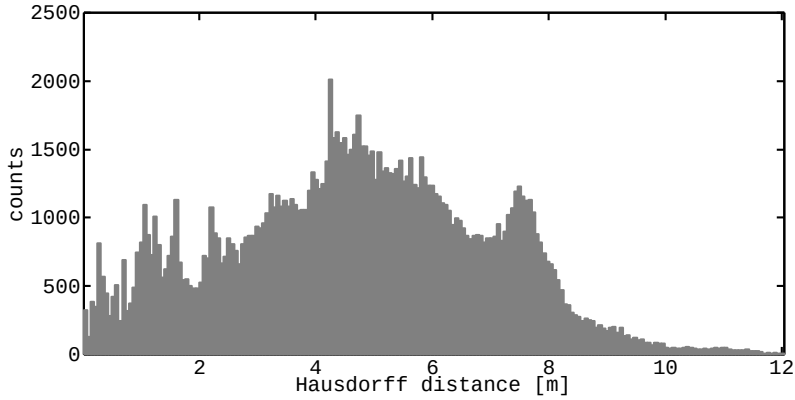
$$\textit{Rayleigh}\,[\sigma] : p_{\textit{Rayleigh}}(d \mid \sigma) = \frac{d}{\sigma^2} \exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{3.29b}$$

If the plan library $\boldsymbol{\mathcal{I}}$ is dense around the robot's future trajectory, with user plans leading forwards and backwards uniformly, then it seems reasonable to expect the coordinates of $\boldsymbol{x}_B - \boldsymbol{y}_B$ in equation 3.27c to be i.i.d. Gaussian with zero mean. Consequently, the Hausdorff distances between the future robot trajectory and the plan library $\boldsymbol{\mathcal{I}}$ will approximately have a Rayleigh distribution.

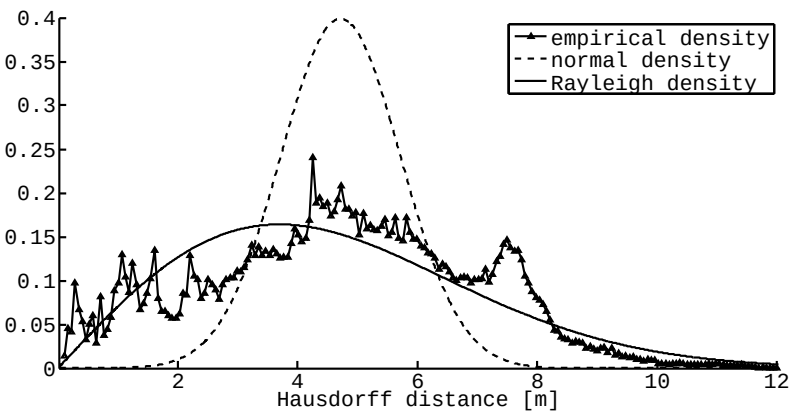
Figure 3.11 shows the histogram, and a number of densities fitted to the complete set of Hausdorff distances of the training set described in chapter 4, section 4.3.4. Figure 3.11(b) compares the empirical density derived from the histogram with the maximum likelihood Rayleigh and Gaussian densities. Clearly, the Rayleigh distribution is the better fit. It is also possible to observe that the empirical distribution is slightly skewed towards larger distance values compared to the Rayleigh distribution because of the hierarchical optimisation steps in equation 3.27.

Distance likelihood. The preceding section uncovers the statistical properties of the resemblance between the robot's future trajectory and all plan libraries $\boldsymbol{\mathcal{I}}$ on a real dataset. However, the distribution of the trajectory metric is not sufficient to calibrate the *plan process probability*, $p_{ppf}\left(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\right)$ because it fails to fulfil the constraints we will present next.

It is still necessary to quantify what the resemblance of a certain plan $\boldsymbol{i}_k$ to the robot's future path *means* in probabilistic terms, and what this information implies for the temporal evolution of user plans. This information is *subjective* because



(a) histogram



(b) densities

FIGURE 3.11: Distribution of the Hausdorff distance for the training set described in chapter 4, section 4.3.4.

it does not follow directly from the statistical properties of the data. Instead, it represents domain knowledge, which we elicit from the properties of the problem. The Bayesian formulation of *plan recognition* of section 3.2.2 essentially allows to fuse such subjective information with other sources of uncertain knowledge in a probabilistically sound way.

Let $\boldsymbol{d}_k = [d_1, \ldots, d_m]^T$ be the observed distances of all user plans at time k to the robot's future trajectory. If the user intends to follow the robot's future trajectory, she will have a plan resembling the future trajectory closest. Therefore, each distance d_j offers an uncertain measurement of the likelihood of each plan $\boldsymbol{i}_k$. We can define the following *distance likelihood*, which relates distance measurements

$\boldsymbol{d}_k$ to the plan library $\boldsymbol{\mathcal{I}}_k$.

$$\text{distance likelihood: } p_{dist}\big(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k\big) \tag{3.30}$$

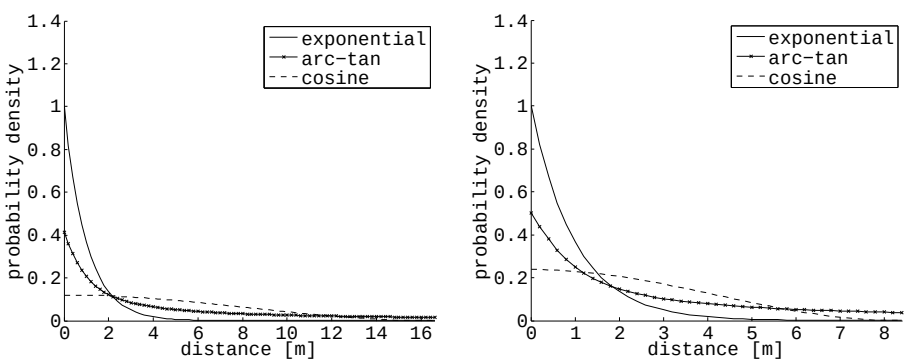
The *distance likelihood* should decrease in function of distance, as far-away plans are less likely to correspond to the robot's future path. With this constraint, we will analyse the performance of three distance likelihoods, the exp-, the arctan- and the cos-likelihoods. Equations 3.31 define the functional form of each likelihood.

$$\text{exponential: } p_{\exp}(d \mid \boldsymbol{i}_k) = \frac{\exp(-d)}{\kappa_{\exp}(d_{max}, d_{min})} \tag{3.31a}$$

$$\text{arc-tangent: } p_{\arctan}(d \mid \boldsymbol{i}_k) = \frac{\frac{\pi}{2} - \arctan(d)}{\kappa_{\arctan}(d_{max}, d_{min})} \tag{3.31b}$$

$$\text{cosine: } p_{\cos}(d \mid \boldsymbol{i}_k) = \frac{1 + \cos\left(\frac{\pi d}{d_{max} - d_{min}}\right)}{\kappa_{\cos}(d_{max}, d_{min})} \tag{3.31c}$$

where $\kappa_{\{\exp,\arctan,\cos\}}(d_{max}, d_{min})$ is a normaliser that ensures that each likelihood obeys the laws of probability. The likelihoods are defined over the range of distances $[d_{min}, d_{max}]$ for the observed robot trajectory. In figure 3.12 we plot the three likelihoods for two distance ranges, a short range and a long range.



(a) long distance range (b) short distance range

FIGURE 3.12: Different distance likelihoods employed in the calibration phase of the plan process function.

The rationale behind introducing a *distance likelihood* is to reduce the amount of data needed to calibrate the *plan process probability*. Normally, in order to

calculate a probabilistic model of plan transitions, one would require sufficient data about *all* possible transitions. The distance likelihood can reduce the amount of training data because information spreads from observed transitions to unobserved transitions between plans. The decay rate of the *distance likelihood* offers a degree of freedom to control this transfer of information.

The decline of the exp-likelihood is the fastest, followed by the arctan- and finally by the cos-likelihood. Therefore, in the plan process function calibrated with the exp-likelihood, few plans surrounding the robot's future path will share all likelihood. Conversely, for the cos-likelihood, likelihood is broadly shared among many plans. The arctan function represents a situation in between both extremes. We will see the impact of this property in the experimental section, section 3.3.

Calibration of the plan process function. The *distance likelihood* imposes soft probabilistic constraints on the user's true intention, which is hidden for the robot. Knowledge about other variables, such as the user's input $\boldsymbol{u}$ can refine the estimation of the user's plan further. Therefore, the *distance likelihood* is a type of uncertain evidence called *virtual evidence* in the terminology of section 2.1.2. It only offers direct, albeit uncertain, information about the user's plan $\boldsymbol{i}_k$. However, this information can propagate inside the Dynamic Bayesian Network, which models the process of plan recognition, to shed light on other quantities such as the *user model*.

Figure 3.13 shows an extended Dynamic Bayesian Network, which can serve to estimate the *local plan process probability*. Please note that the *distance likelihood* is only available during training, when the robot's full trajectory is known. In operation, the robot lacks access to its future path.

The *distance likelihood* imposes a series of soft constraints, which allow to solve for the parameters of the Dynamic Bayesian Network in figure 3.13. These parameters are the *plan process probability*, and if the user interface is discrete, also the *user model*. User models for continuous interfaces are discussed in section 4.3.

We compute the solution for the *plan process probability* as the Maximum A Posteriori (MAP) distribution obtained by the Expectation Maximisation (EM) algorithm (Dempster et al., 1977). The exact procedure for a Bayesian Network is described in (Heckerman, 1998). In broad terms, during the Expectation (E) step the expected sufficient statistics of each component distribution, $p(A_i|parents(A_i))$, of the Bayesian Network are calculated, by propagating evidence with for instance, the Junction Tree Algorithm (section 2.1.3). Then, in the Maximization step (M), the expected sufficient statistics are assumed to be the actual statistics, which allow to compute each component's MAP distribution.

The MAP distribution allows to place a prior on the learning procedure that ensures unobserved training data are still possible. In this case, the prior over user plans at time k vanishes as a uniform distribution during the *plan-recognition time update* (cf. equation 3.20). The posterior depends then solely on the user model. If the

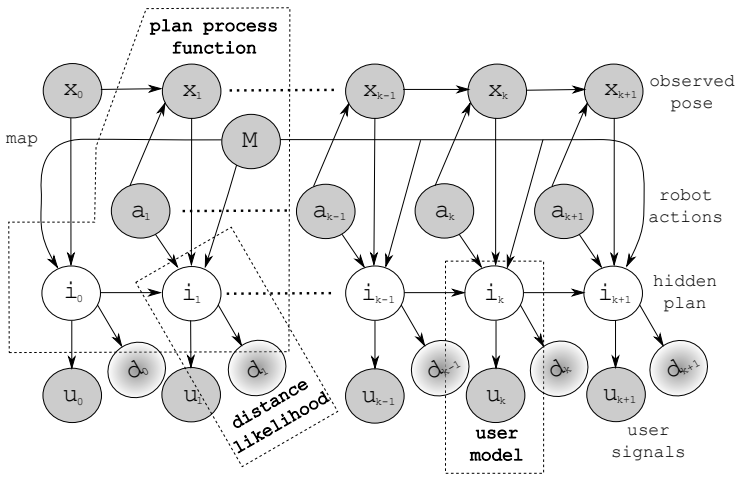


FIGURE 3.13: Dynamic Bayesian Network employed to acquire the parameters of the plan process function and user model. In the calibration phase, the future path followed by the robot can be employed as an indication of the hidden user plan. This observation is incorporated into the Bayesian Network via the distance likelihood of the observed node $\boldsymbol{d}_k$. The distance nodes $\boldsymbol{d}$ appear in light grey because they represent uncertain evidence. Conversely, other variables such as the user's input $\boldsymbol{u}$ or robot actions $\boldsymbol{a}$ are known with absolute certainty and are coloured in dark grey.

robot were using the Maximum Likelihood *plan process probability*, unobserved plan transition during training would always be impossible in operation.

We implemented the learning approach for the plan process probability with the Bayesian Network Toolbox (Murphy, 2010). An extension was necessary to handle uncertain evidence during the EM algorithm.

3.2.5 Summary of the models for plan-recognition

This section summarises all probabilistic models for *plan recognition* we have introduced previously and cites their most relevant properties. The details correspond to the actual implementation and the learning procedures developed in the thesis.

Before stating the properties of the models for plan recognition we define two probability distributions. The Multinomial is a probability distribution over the counts of a variable $\boldsymbol{x}$, which can be in one of K mutually exclusive discrete states,

each with probability p_i. Let m_i denote the counts of state i, then:

$$\mathit{Multinomial}\big(\boldsymbol{m} \mid N, \boldsymbol{p}\big) = \frac{N}{\prod_{i=1}^{K} m_i!} \prod_{i=1}^{K} p_i^{m_i} = \binom{N}{m_1 \dots m_K} \prod_{i=1}^{K} p_i^{m_i} \tag{3.32}$$

where $0 \leq p_i \leq 1$ and $\sum_{i=1}^{K} p_i = 1$ because $\boldsymbol{p}$ are probabilities. $N = \sum_{i=1}^{K} m_i$. The Dirichlet is a probability distribution over a set of K probability values $\boldsymbol{p}$.

$$\mathit{Dirichlet}\big(\boldsymbol{p} \mid \boldsymbol{\alpha}\big) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} p_i^{\alpha_i - 1} \tag{3.33}$$

where $\alpha_i > 0$. The Gamma function is defined as $\Gamma(x) \equiv \int_0^\infty u^{x-1} e^{-u} du$.

TABLE 3.1: User model

symbol	$p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k)$ or $p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{i}_{k-1})$
description	The user model represents the likelihood of current user input signals $\boldsymbol{u}_k$, given the user has plan $\boldsymbol{i}_k$ in mind. It reflects how a specific user translates her plans into inputs to the robot. The user model must be tailored to the particular driving style of each individual.
domain of the variables	User inputs $\boldsymbol{u}_k$ can be either continuous or discrete. *Continuous* inputs correspond normally to the linear and angular velocities of the robot. *Discrete* user inputs represent discrete categories of turning left, right, or of advancing forwards and backwards. User plans $\boldsymbol{i}$, although continuous, must belong to the finite plan library $\boldsymbol{\mathcal{I}}$.
distribution	Depends on the type of user input device. • For *discrete* input devices, the user model has a Multinomial distribution. • For *continuous* input devices, the user model is a Gaussian Process.
calibration	Depends on the type of user input device. • For *discrete* input devices, the procedure is the same as for the plan process function, which we describe in the following table. • For *continuous* input devices, please refer to section 4.3.
prior	Depends on the type of user input device. • For *discrete* input devices, the prior is a Dirichlet distribution with Equivalent Sample Size (ESS) of $\frac{1}{\sharp(\boldsymbol{\mathcal{I}}_k)}$ or $\frac{1}{\sharp(\boldsymbol{\mathcal{I}}_{k-1})\cdot\sharp(\boldsymbol{\mathcal{I}}_k)}$ The ESS is a fictional dataset modelling the confidence in the prior (Heckerman et al., 1995). With the ESS defined previously each cell of the user-model prior gets initialised with $\frac{1}{\sharp(\mathcal{U}_k)\cdot\sharp(\boldsymbol{\mathcal{I}}_k)}$ or $\frac{1}{\sharp(\mathcal{U}_k)\cdot\sharp(\boldsymbol{\mathcal{I}}_{k-1})\cdot\sharp(\boldsymbol{\mathcal{I}}_k)}$. • For *continuous* interfaces, the prior is a Gaussian Process (cf. section 4.3).

TABLE 3.2: Plan process function

symbol	$p_{ppf}\big(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M}\big)$
description	The plan process function is a probabilistic model of the temporal transition between user plans. It reflects how information from the past propagates to the present. The plan process function is user-independent.
domain of the variables	User plans $\boldsymbol{i}$, although continuous, must belong to the finite plan library $\boldsymbol{\mathcal{I}}$. Robot actions $\boldsymbol{a}_k$ belong also to a discrete set, but can be continuous. The plan process probability is not parametrised over possible maps and robot poses. The dependency on $\boldsymbol{x}_k$ and $\boldsymbol{M}$ only reflects that the user plans in the plan libraries $\boldsymbol{\mathcal{I}}_{k-1:k}$ must be adapted to the environment.
distribution	Multinomial distribution in $\boldsymbol{i}_k$ parametrised by the discrete set of prior user plans $\boldsymbol{i}_{k-1}$ and robot actions $\boldsymbol{a}_k$.
calibration	We calculate the MAP distribution for the plan process function through the Expectation Maximisation (EM) algorithm for Dynamic Bayesian Networks.
prior	Uniform prior during calibration. Such prior takes the form of a Dirichlet distribution with Equivalent Sample Size (ESS) of $\frac{1}{\sharp(\boldsymbol{A})\cdot\sharp(\boldsymbol{\mathcal{I}}_{k-1})}$.

TABLE 3.3: Distance likelihood

symbol	$p_{dist}\big(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k\big)$
description	The distance likelihood models the likelihood of user plans $\boldsymbol{\mathcal{I}}_k$, given their distances to the robot's future trajectory $\boldsymbol{x}_{k:end}$.
domain of the variables	$\boldsymbol{d}_k = [d_1, \ldots, d_m]^T$, where each $d_i \in [d_{min}, d_{max}]$. $\boldsymbol{\mathcal{I}}_k$ is the plan library at time k comprising a finite set of user plans.
distribution	exp-likelihood, arctan-likelihood and cos-likelihood in equation 3.31. Other decaying distributions over $[d_{min}, d_{max}]$ are possible.
calibration	This model is domain knowledge, which we elicit from the properties of the calibration procedure for the plan process function.
prior	Not applicable.

3.2.6 History of the framework for plan recognition

The preceding sections explain the process of recognising the user's plans. The goal of this section is to provide a historical overview of the framework's evolution.

The problem of plan recognition in the context of joint human-robot wheelchair navigation was first proposed in (Demeester et al., 2003). The authors demonstrated on a simulated wheelchair how to recognise the user's intention to drive to a certain goal location. In this early work, user intentions were simply end poses to be reached at an end velocity. Later work (Demeester et al., 2006; Demeester et al., 2008) recognised the need to include also the user's plan to reach a goal location in the intention representation, as users might reach the same goal location in different ways. This work still employed a global plan process function, and the experiments were performed for an able-bodied user on a real wheelchair. The user model for this case is discussed in section 4.3.1. At the same time, (Huntemann et al., 2007a) investigated the impact of driving disabilities on the user model, but still considering global trajectories. Afterwards, we decided to change the representation of user plans to local trajectories, since immediate assistance should rely primarily on local information and global maps are not always available. In (Huntemann et al., 2008a) we describe a learning technique to acquire user models based on local trajectories for arbitrary driving styles. The contribution of this chapter is to extend the results of (Huntemann et al., 2008a) by completing the process of plan recognition with local trajectories.

3.2.7 Related work

Assistive robotics. The software architecture of nearly all existing robotic wheelchairs, robotic guide dogs for the visually impaired, and robotic walking assistants is comprised of several *modes*, sometimes called *behaviours* or *agents*. Among these modes, the *user-control mode* is the most common one. In principle, plan recognition is not required in user-control mode, but it may be useful for suggesting assistive actions. For example, traditional scanning interfaces for wheelchairs successively highlight one of a set of elementary directions the driver may want to take, such as *turn left* or *drive forward*. With a binary yes/no signal, the user can indicate if she actually wants the wheelchair to follow the direction that is highlighted at that time instant. If these commands were augmented with a more sophisticated task that the user is estimated to desire to execute, the chair's functionality would be enhanced without requiring additional physical capabilities of the user. This may considerably reduce the travel time and increase safety. Besides a user-control mode, an *autonomous* mode is often defined, in which user signals are completely disregarded and plan recognition amounts to an explicit indication by the user of the goal location that she would like to reach. Many assistive robots also offer several *semi-autonomous* modes that require continuous collaboration between robot and user. Each of these semi-autonomous modes encodes a separate user intent. Various projects adopt modes that are directly derived from the

specification of tasks that a wheelchair driver typically wants to perform, such as *avoid obstacle*, *drive through door*, or *dock at table* (Simpson et al., 1998; Röfer and Lankenau, 2000). Others specify modes based on environmental characteristics, such as *outdoor* versus *indoor* (Yanco, 2000), *cluttered* versus *open*, or *dynamic* versus *static* (Prassler et al., 1999).

The presence of multiple modes requires a selection or combination procedure for these modes. This corresponds to the plan recognition problem. Some systems require the user to select the correct mode *manually* (Prassler et al., 1999) or (Parikh et al., 2004). However, this work focuses on robotic assistance that activates modes automatically, as these are hoped to be less tiring and more intuitive than manual mode selection procedures. Various approaches exist for this *automatic* mode selection. One option is to merge the motion commands given by several assistance modes, though this is a less frequent approach (Aigner and McCarragher, 2000). Another option is to select a single mode to be active, resulting in a *winner-takes-all* strategy, which is the prevailing strategy. Most approaches can be considered to determine how *likely* or *suitable* a certain assistance mode is at a certain time instant, after which the shared control algorithm selects the assistance mode that is most suitable. This suitability measure, though it is not always explicitly calculated, may be based on several possible information sources:

1. **Sensor signals**. Some suitability values of assistance modes are purely based on sensor information. These approaches disregard the user signals, and are therefore better classified as fully autonomous systems. (Simpson et al., 1998) adopt a Bayesian network that uses evidence from the latest sensor readings and from an estimate of the robot location to calculate the probabilities of assistance modes *follow corridor*, *drive through door*, and *avoid obstacle*. (Yanco, 2000) adopts several sensors and a trained classifier to detect whether the wheelchair is outside or inside, and to trigger the corresponding outside/inside assistance mode. However, disregarding user signals may activate incorrect assistance modes. Consider for example a wheelchair user standing near a door in a corridor. Intuitively, a decision to drive through the doorway should stem from the user pointing more or less in the direction of the door, and if not, the user may be assumed to desire to follow the corridor. If only sensor signals are used however, the system would e.g. always opt to drive through the doorway independently of the user signals, strongly reducing the user's freedom in this way.

2. **Sensor and user signals**. Other automatic mode selection techniques base their decisions both upon environmental perception and user signals. Though these approaches adhere more to the user's expectations as they consider user signals to activate certain assistance modes, their user model is usually fixed and hard-coded, i.e. the same decision rules are adopted for all drivers. Examples of these approaches can be found in (Special Issue RAM, 2001; Mittal et al., 1998).

3. **Sensor signals, user signals, and assistance actions**. Very few suitability values are calculated by additionally taking the outputs of assistance modes into account (Aigner and McCarragher, 2000; Parikh et al., 2004). (Aigner and McCarragher, 2000) compare the user signals with the outcome of an avoid-obstacle assistance mode in order to decide which commands should be sent to the motors. In case of a strong conflict between these signals, the user obtains full control, but at reduced velocity.

Hence, an important difference with the framework proposed in this chapter is that most existing approaches determine the most likely assistance mode using the latest sensor and user interface signals only, and uncertainty upon these signals is often disregarded. Moreover, due to the rather broad specification of assistance modes, a more accurate estimation of the user's plan has to be performed by these assistance modes once they are activated. For example, activating the obstacle-avoidance mode does not yet specify whether to avoid an obstacle to the left or to the right, and to which extent the obstacle should be avoided to the left or to the right. It is the responsibility of the assistance modes to infer these more concrete user plans. In contrast, we proposed a framework for plan recognition that can be tailored to the particular driving characteristics of the user. This framework explicitly deals with uncertainty present in user signals and user plans, and merges past beliefs regarding user plans with new evidence from user signals.

Plan recognition Approaches that are comparable to our probabilistic plan recognition framework can be found in the analysis of motion behaviours. (Bennewitz et al., 2005; Bennewitz et al., 2002) propose a framework to cluster and predict human motion based on past motion. Their system classifies walking trajectories into motion behaviours by the Expectation Maximization (EM) algorithm. The goal of their framework is to find the Maximum a Posteriori motion pattern given a trajectory in order to predict the motion of people.

(Glover et al., 2004) estimate the activity a person is engaged in when using a walking aid. They adopt a Hidden Markov Model (HMM) that integrates metric, topological and temporal information into a probabilistic estimate over activities. In contrast to the framework proposed in this work, neither of these approaches directly includes user signals into predictions of motion behaviour or motion plans, and they implicitly assume that users can go everywhere autonomously.

3.3 Experimental results

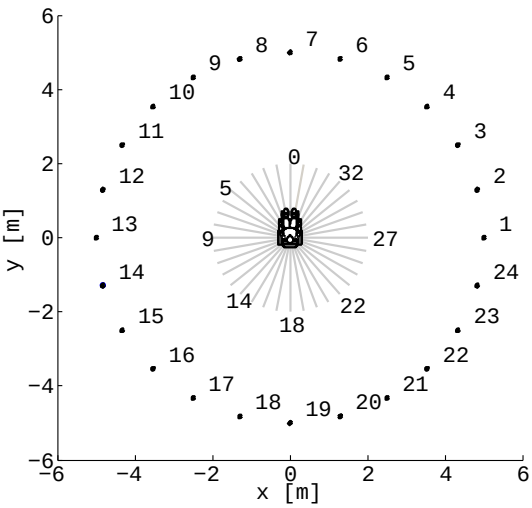
3.3.1 Plan recognition for a discrete user interface

The goal of this section is to illustrate the process of plan recognition on a simplified example, which is however general enough to draw meaningful conclusions. Figure 3.14 shows a simulated wheelchair in an obstacle-free environment. The task of the user consists in reaching a set of goal positions in random order. In order to do so, the user can steer the wheelchair with the discrete interface shown in figure 3.14(b).

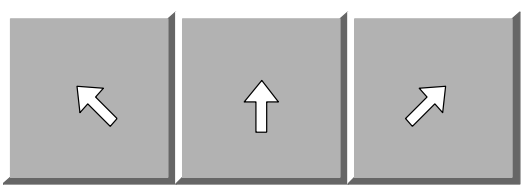
Although the user can only advance forwards or turn in discrete steps, such interface already allows to navigate in everyday environments. For instance, (Galán et al., 2008; Vanacker et al., 2007; Demeester et al., 2009b) report navigation experiments in simulation and with a real wheelchair controlled by a Brain Computer Interface (BCI). Essentially, BCI corresponds to a three button interface, where observations of navigation commands are noisy. In BCI, the system classifies the user's brain-waves, while she performs a set of distinguishable mental tasks, into a discrete set of classes. It is possible to incorporate a BCI interface in the Dynamic Bayesian Network of figure 3.2 by adding an additional node for the observation likelihood, $p_{obs}(\boldsymbol{b}_k|\boldsymbol{u}_k)$, which maps observed brain signals $\boldsymbol{b}_k$ to unobserved inputs $\boldsymbol{u}_k$.

Centred on the wheelchair in figure 3.14(a) we show a template of 36 local paths of limited extension. They are numbered in counter-clockwise order and always relative to the wheelchair's current orientation. These paths are recomputed whenever the wheelchair moves, but bear no relation with the set of goal locations. The question we are trying to address in the ensuing discussion is whether these local trajectories suffice to predict the plan of the user, and whether it is possible to perform such predictions across time without reference to any global location.

We will study plan recognition by means of the Dynamic Bayesian Network of figure 3.4. The user signals in this DBN depend both on the current and prior plans, which introduces a one-step memory in the plan recognition process. Modelling the user's interaction with the wheelchair requires to estimate the parameters of this DBN. These parameters are the user model, $p_{user}\left(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{i}_{k-1}\right)$ and the plan process function, $p_{ppf}\left(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k\right)$. Since all variables are discrete, the user model and plan process function can be learned simultaneously in the same framework. Therefore we will discuss both models together. In chapter 4.3 we propose a separate user modelling technique for continuous joysticks. The general idea of the plan process function with discrete plans and actions generalises however to the continuous interface as we will show in section 3.3.2.



(a) training environment



(b) user interface

FIGURE 3.14: Wheelchair moving in free space controlled by a discrete interface. The user has to move to various goal locations in random order. During driving her plan is estimated based on a set of local linear paths.

Plan recognition models for large displacements calibrated with global paths

In this section we report the results on user modelling and plan recognition for a discrete user interface which allows large displacements. If the user points forwards, the wheelchair advances $1m$, and whenever she selects actions left or right, the wheelchair turns $0.5rad \approx 30^o$.

The user model, $p_{user}\left(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{i}_{k-1}\right)$, represents the likelihood of the user input, $\boldsymbol{u}_k$, provided the user has a certain plan in mind. In this case, the input depends on current and past plans in order to stabilise changes in user plans. Further, this input is observed, but user plans are unknown. Similarly, the plan process function, $p_{ppf}\left(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k\right)$, models the evolution of unobserved user plans if the robot selects action $\boldsymbol{a}_k$.

User plans can be estimated *a posteriori* from the user's future trajectory. If the user is in full control, a sequence of inputs will result in a certain intended trajectory. It is possible to detect driving disabilities or misconceptions about wheelchair dynamics if the user is instructed to complete a set of known tasks. Deviations from the prescribed task concern the user model, which reflects how user plans translate into wheelchair inputs. The plan process function, on the other hand, represents the evolution of plans depending on the actions of the robot, regardless of any user input.

During calibration of user model and plan process function, we segment the user's trajectory according to a sequence of goal locations that the user should reach. At every time step, the future path until the next goal determines ground-truth. There are two options to estimate the true plan out of a plan library such as in figure 3.14(a). The first approach selects the plan that approximates ground-truth most accurately and employs this plan for calibration. This option requires large quantities of training data since transitions between all possible hypotheses must be considered. Alternatively, the assignment of plans to ground-truth can obey soft constraints. We formalise these soft constraints as an observation likelihood, $p_{dist}(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k)$, which captures how each plan hypothesis represents ground truth. The distance likelihood, $p_{dist}(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k)$, provides an initial guess about user plans based on the distance to ground truth. This likelihood is only available in the calibration phase, when the complete trajectory of the user is available. We will compare the three different distance likelihoods shown in figure 3.12, namely the exp-likelihood, the arctan-likelihood and the cos-likelihood. All three likelihoods decrease as a function of distance to ground-truth, but at different rates. The advantage of enforcing soft constraints is that information from a reduced set of driving examples can propagate to unobserved hypotheses. Care must be taken, however, to ensure that inferring the user's plan is still possible despite the additional source of noise.

Figure 3.15 shows the user model and posterior distributions calibrated with the exp distance-likelihood. The user model in figure 3.15(a) (front view) and figure 3.15(b) (top view), reveals that "left" signals are likely whenever the user's current plan,

$\boldsymbol{i}_k$, is in the range of 0° to 180°. We will identify this region as the *left half-space* because it corresponds to plans on the left side of the wheelchair. The user points "right" for current user plans in the range of 180° to 350°, which are the plans on the wheelchair's right side. Analogously, we will call this region the *right half-space*. User input "forward" is likely for current plans close to the origin, either at the left side (range 0° to 50°) or at the right side (310° to 350°). We do not consider orientation 360° as it coincides with orientation 0^0.

The top view of the user model in figure 3.15(b) allows an even finer analysis. We have drawn a solid line of unit slope in the space of current ($\boldsymbol{i}_k$) and prior ($\boldsymbol{i}_{k-1}$) user plans. Please note that the user's current plan is represented on the x-axis. In this plot the patches of white colour correspond to a uniform user model and all signals are equally likely.

"Left" signals are likely for prior plans $\boldsymbol{i}_{k-1}$ above the unit slope separation in the left half-space. By turning left, the user intends to reduce her misalignment with a goal location, when her prior plan is in the range of 0° to 180°. The purpose is to reach a final orientation of 0° in correct alignment with the user's current goal along decreasing orientations.

"Right" signals are likely for prior plans below the separating line in the right half-space. Reducing the misalignment to a goal by turning right involves increasing current plan orientations starting above 180° until reaching the correct alignment at 360°. Interestingly, in the right half-space, below the separator, the likelihood clusters around independent likelihood "islands". This effect can be attributed both to the fast decay of the exp-distance-likelihood as well as to the large change in orientation when turning, i.e. a single "right" signal changes the wheelchair's heading 30°.

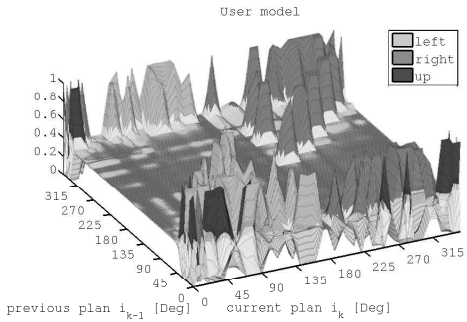
Another interesting part in the user model concerns the higher likelihood in the bottom and top part of figure 3.15(b), that is for prior plans $\boldsymbol{i}_{k-1}$ on the left side (range 0° to 50°) and right side (310° to 350°) of the wheelchair. In these regions, the likelihood is clearly split among left and right turns, where the likelihood for each input is higher in its respective half-space. Input "forward" is likely in the corners of figure 3.15(b). For all user inputs, this part of the user model represents the final approach phase when reaching a goal. During the approach, the wheelchair is relatively aligned with the goal orientation. Hence the prior plan focuses narrowly around a small area. Once the user reaches her destination, the next goal's orientation can be located at any orientation, which causes the user model to spread over the complete range of current plans. Acting rationally, this user chooses "left" or "right" input signals depending on the relative location of the next goal.

Figure 3.15(c) and figure 3.15(d) show the posterior distributions as a function of the current user plan, $\boldsymbol{i}_k$. These posteriors are the result of combining user model and plan process function during several iterations. User signal "left" makes posterior plans on the left half-space, i.e. for relative orientations between 0° and 180°, more probable. Conversely, the posterior probability is highest in the

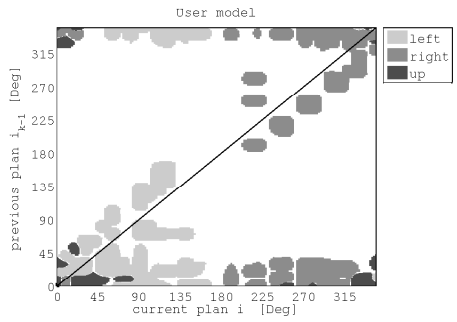
right-half space if the user issues "right" signals. An interesting property of the posterior plots for "left" and "right" signals in figure 3.15(c) is the presence of maxima spaced approximately according to the angular displacement at which the wheelchair turns. Further, repeated iteration of signal "left" shifts posterior probability mass to the left, i.e. to lower relative orientations. Likewise, multiple "right" signals shift the peaks of the posterior to higher relative orientations at the right. The posterior probability for user signal "forward" is highest in the proximity of relative orientation 0°, at both sides of the left-right divide. Repeated iterations of signal "forward" increase the posterior probability of the plan pointing straight-ahead, at the expense of plans slightly pointing left or right.

Figures 3.16 shows the user model and posterior distributions calibrated with the arctan-distance-likelihood. The structural properties of the user model and posterior are similar to the results obtained for the exp-distance-likelihood. The user model in figure 3.16(a) (front view) and in figure 3.16(b) (top view) is again divided into left and right half-spaces with "left" and "right" user signals more likely in their respective half-spaces. Further, "left" signals are likely for prior plans $\boldsymbol{i}_{k-1}$ above the unit slope separator, and "right" signals are likely for prior plans below the separator (figure 3.16(b) top view of the user model). Again, in the top view of the user model it is possible to distinguish two high-likelihood stripes in the top and bottom part of figure 3.16(b). These likelihoods correspond also here to the data obtained while approaching a goal location. The arctan-distance-likelihood results differ from the exp-distance-likelihood in the way information spreads to unseen training examples. The user model plots show broader peaks for the arctan-likelihood. The posterior plots for "left"- and "right"- signals in figure 3.16(c) show also a pattern of maxima spaced broadly according to the change in heading of the corresponding action, but the peaks are broader. Repeated iteration of signals "left" or "right" shifts posterior probability to lower or higher plan numbers respectively. The posterior of user signal "forward" in figure 3.16(d) is maximal around the relative orientation 0° and plans pointing slightly left (orientations close to 30^0) and right (orientations close to 350^0). Repeatedly issuing signal "forward" reduces the posterior probability of left and right plans in favour of the straight-ahead direction.

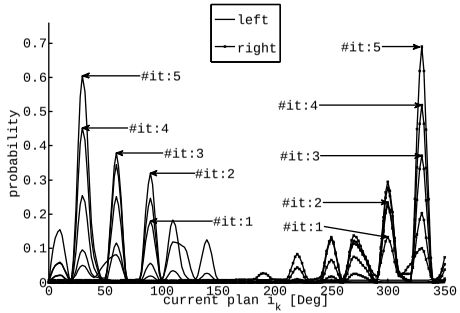
The user modelling and plan recognition results for the cos-distance-likelihood are reported in figure 3.17. These plots preserve the overall structure of the exp-likelihood results, albeit with a larger level of noise. In the top view of the user model, figure 3.17(b), the split in left- and right half-spaces is still apparent. However, signal "right" is also likely in the left half-space, and vice versa. Signal "forward" is likely in a wider area than for the user model calibrated with the exp- or arctan-distance-likelihoods. The top and bottom stripes corresponding to the final approach phase have expanded as well. Concerning the posterior distributions, we observe a higher level of noise, particularly for signal "left" in figure 3.17(c). The posterior for signal "right" maintains a similar distribution to the exp-distance likelihood results of figure 3.15(c). However, the spacing of peaks is less uniform.



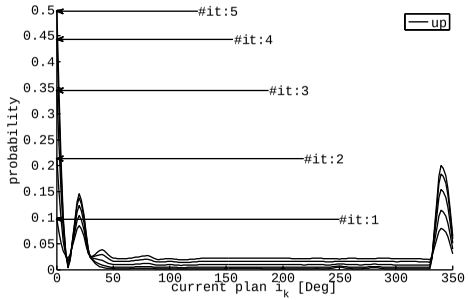
(a) front view user model



(b) top view user model

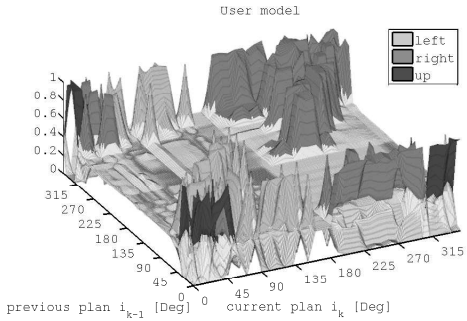


(c) posterior for action "left" and "right"

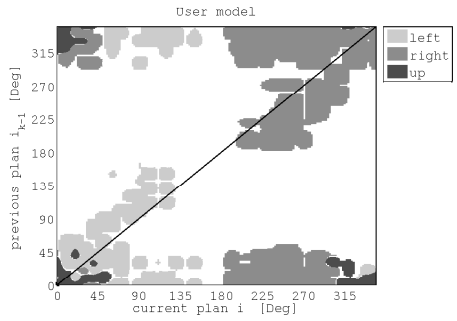


(d) posterior action "forward"

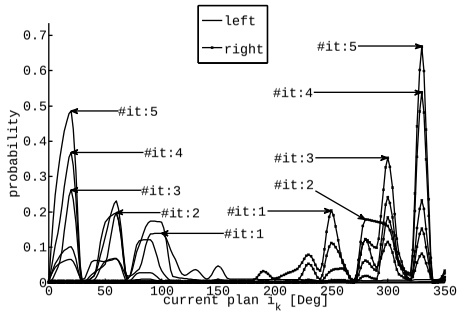
FIGURE 3.15: User model and posterior distributions of a discrete interface calibrated with the exp-distance-likelihood by employing the full path to the next global intention, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.



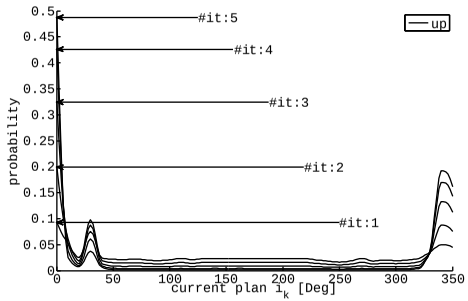
(a) front view user model



(b) top view user model

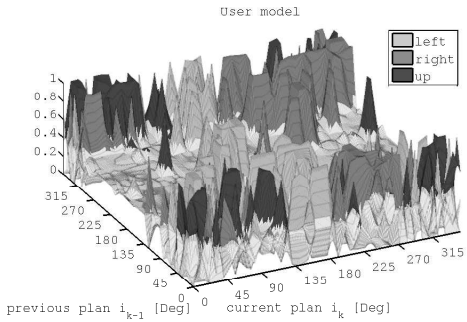


(c) posterior for actions “left” and “right”

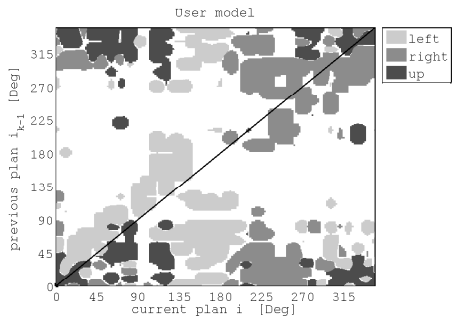


(d) posterior action “forward”

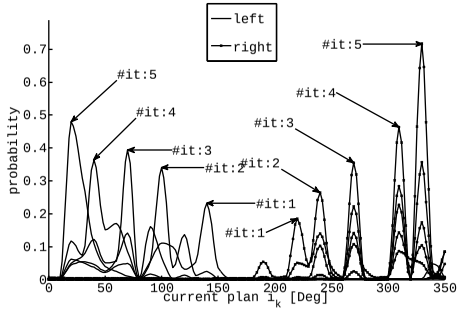
FIGURE 3.16: User model and posterior distributions of a discrete interface calibrated with the arctan-distance-likelihood by employing the full path to the next global intention, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.



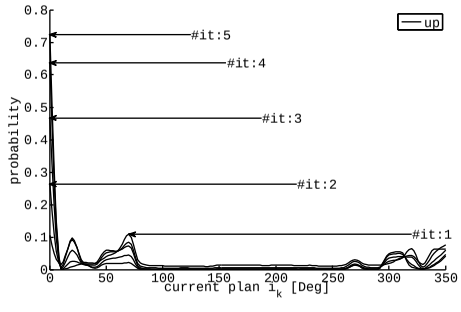
(a) front view user model



(b) top view user model



(c) posterior for actions “left” and “right”



(d) posterior for action “forward”

FIGURE 3.17: User model and posterior distributions of a discrete interface calibrated with the cos-distance-likelihood by employing the full path to the next global intention, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.

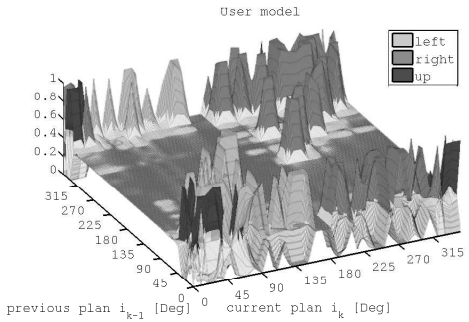
Plan recognition models for large displacements calibrated with local paths

In this section we report comparable user modelling and plan recognition results to the previous section for the exp-, the arctan-, and the cos-distance-likelihoods. The robot is controlled with the same discrete interface, which moves the wheelchair forward for $1m$ and turns it left or right for $0.5rad \approx 30^o$. Both sections differ, however, in the specification of ground-truth, which is employed to calibrate the plan recognition models. Instead of considering the full path to a global destination, the distance likelihood $p_{dist}(\boldsymbol{d}_k \mid \boldsymbol{\mathcal{I}}_k)$ relies only on a local segment of limited extension. The purpose is to obtain a local approach with as little global information as possible. The only remaining global information imposes that the path of limited extension cannot exceed the next goal location when the user is still trying to reach it. Otherwise, the calculated plan process function would consider transitions between plans which do not correspond to intended behaviour. The user completes a task by reaching the goal location prior to changing her destination. These global locations could be replaced in the future by locations where the user stands still for a minimum amount of time.

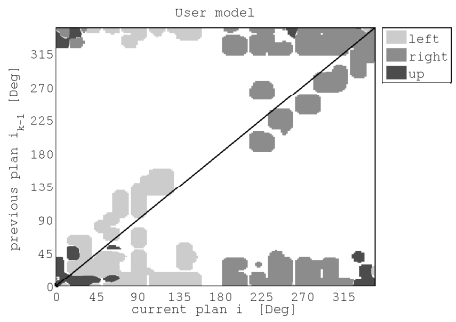
The main properties described in the previous section still hold for the local approach. The exp-distance-likelihood (figure 3.18) leads to models with the highest concentration of probability mass around observed examples. In the top view of the user model, figure 3.18(b), "left" signals are again more likely in the left half-space when the plan number decreases from prior to current plans. The "left" signal's likelihood is highest above the separator marked with a solid line. Symmetrically, "right" signals are likely below the separator whenever the user turns to increase her relative orientation to a goal, starting around 180° to arrive at 360°. "Forward" signals are likely for small orientation differences to a goal. The remaining likelihood corresponds also here to the final approach phase, when the user is relatively aligned with a goal before reaching it and changing afterwards to a new task. The posterior for "left" and "right" signals in figure 3.18(c) presents the characteristic pattern associated with the large orientation changes of a turn.

The arctan-distance-likelihood results are comparable to the exp-likelihood, except for slightly broader peaks in the user model of figures 3.19(a) (front view) and 3.19(b) (top view). The slower decrease rate of the arctan-distance-likelihood causes information to transfer to unobserved training examples. A posteriori, repeated "left" signals shift probability mass to lower plan orientations in figure 3.19(c), whereas repeated "right" signals shift probability in the opposite direction. This shift occurs along evenly spaced peaks with a peak-to-peak angular distance matching approximately the discretised turning angle.

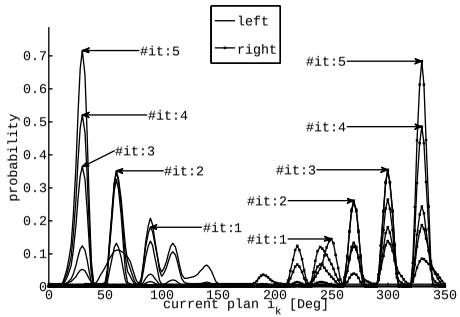
The cos-distance-likelihood (figure 3.20) yields fairly similar results to the arctan-likelihood. The peaks of the user model in figures 3.20(a) and 3.20(b) are only slightly broader and the maxima in the posterior of figure 3.20(c) are less evenly spaced. Such similarities can be attributed to the resemblance of both distance likelihoods in the local range (figure 3.12(b)), especially at low distances.



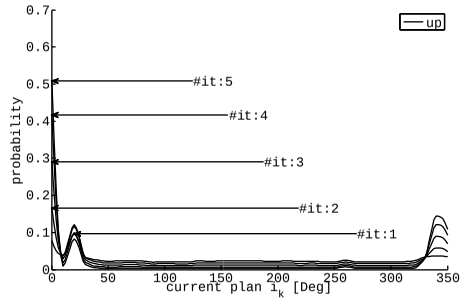
(a) front view user model



(b) top view user model

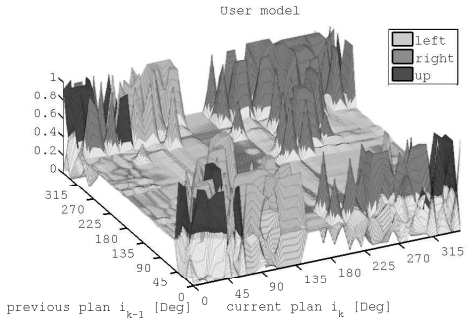


(c) posterior for action "left" and "right"

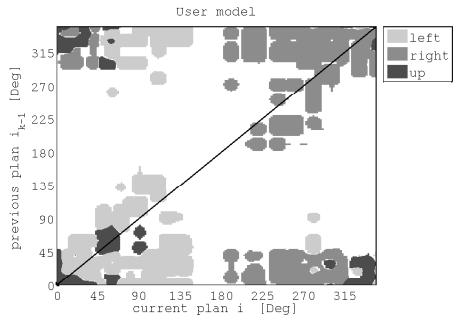


(d) posterior for action "forward"

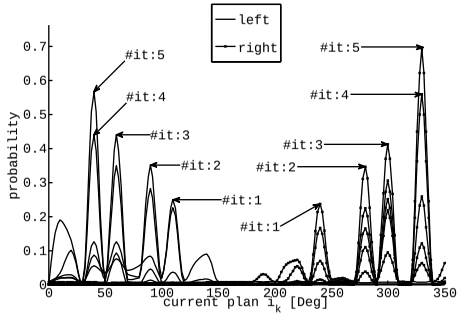
FIGURE 3.18: User model and posterior distributions of a discrete interface calibrated with the exp-distance-likelihood by looking only a maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.



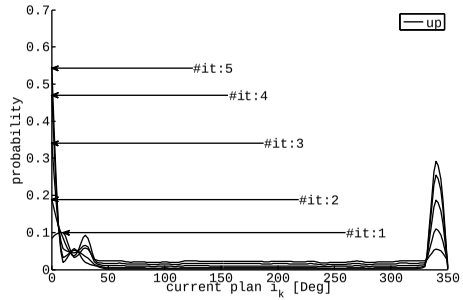
(a) front view user model



(b) top view user model

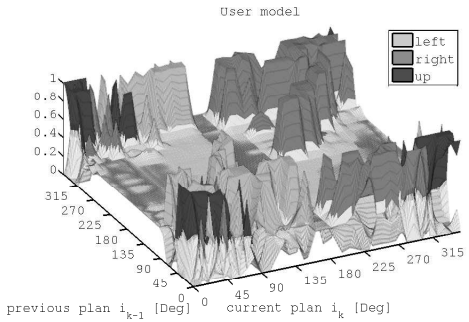


(c) posterior for action “left” and “right”

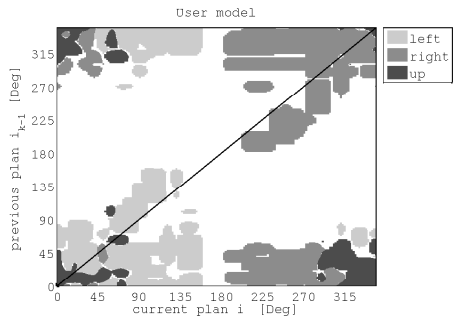


(d) posterior for action “forward”

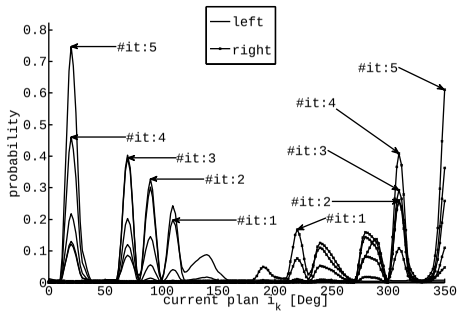
FIGURE 3.19: User model and posterior distributions of a discrete interface calibrated with the arctan-distance-likelihood by looking only a maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.



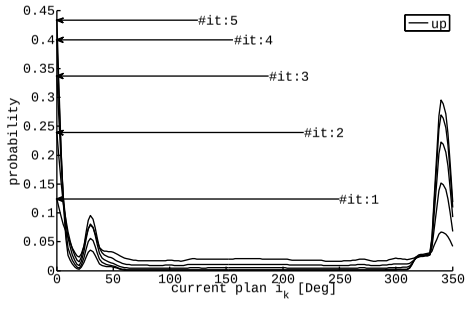
(a) front view user model



(b) top view user model



(c) posterior for action "left" and "right"



(d) posterior for action "forward"

FIGURE 3.20: User model and posterior distributions calibrated with the cos-distance-likelihood by looking only a maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (1m, 0.5rad)$.

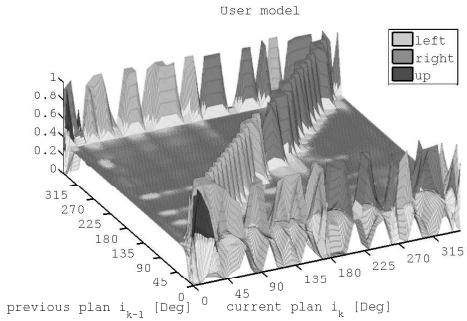
Plan recognition models for small displacements calibrated with local paths

Next we analyse the impact on plan recognition and user modelling when the displacements associated with the discrete interface are smaller. In this case, signal "forward" advances the wheelchair $0.2m$ and signals "left" or "right" changes the wheelchair's orientation $0.1rad \approx 5.73^o$. The plan recognition models are calibrated also here following a local approach, i.e. with paths of limited extension that respect the sequencing of global destinations.

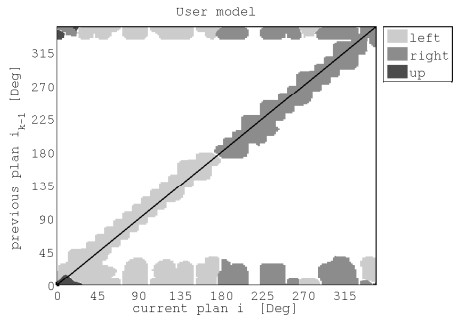
The same structure of the user model obtained for large displacements is still recognisable in figures 3.21(a) and 3.21(b). For the exp-distance-likelihood, the current plan $\boldsymbol{i}_k$ divides the user model in left- and right half-spaces. "Left" signals are more likely for current plans below relative orientation 180^0 in the left-half-space. Likewise, "right" signals are likely in the right-half-space for current plans $\boldsymbol{i}_k$ above relative orientation 180^0. We have highlighted in the top view (figure 3.21(b)) a separator of unit slope between current and previous plans. "Left" signals are more likely above the separator, and "right" signals below. In the former case, the user reduces the orientation error along decreasing plan numbers, and in the latter case, by turning right, the user intends to increase the wheelchair's orientation until aligning with the goal at 360°. The user models of large and small displacements differ, however, in the spacing of likelihood peaks. For small displacements, the user model is composed of narrow contiguous peaks, whereas for large displacements, the user model is more sparse and the likelihood peaks are broader. This pattern is also apparent in the segment of the user model corresponding to the final approach phase, at the top and bottom of figure 3.21(b). The posterior distribution for "left" and "right" signals (figure 3.21(c)) no longer comprises equispaced maxima and the distribution is more uniform. However, repeatedly turning left or right still shifts posterior probability in the correct direction.

Figure 3.22 shows the user model and the posterior for the arctan-distance-likelihood. Due to the lower decrease rate of the arctan-distance-likelihood, the peaks in the user model of figure 3.22(a) are wider than for the exp-likelihood. "Left"- and "right" signals still divide the user model into two half-spaces, but the frontier marked by the unit slope separator in the top view of figure 3.22(b) is no longer clear. The likelihood leaks through the frontier for both signals. The posterior distribution for "left" and "right" signals (figure 3.22(c)) is more uniform than its counterpart involving larger displacements, since small displacements imply smoother transitions between plans. Some irregularities of the user model transfer to the posterior in figure 3.22(c). For instance, the indentation in the posterior for relative orientation 290^0 matches an irregular region in the user model.

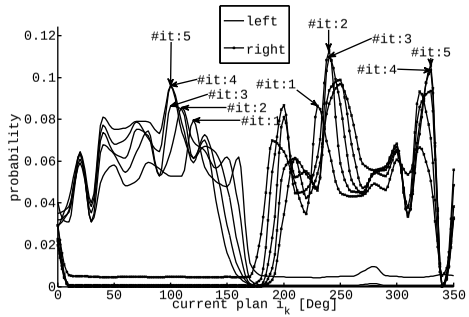
The results for the cos-distance-likelihood in figure 3.23 involve higher levels of information transfer to unseen examples than for exp- or the arctan-likelihoods respectively. The top view of the user model, figure 3.23(b), reveals broader regions where the discrete user signals are likely. Further, the posterior for "left" and "right" signals (figure 3.23(c)) is also more uniform than its counterpart for larger displacements.



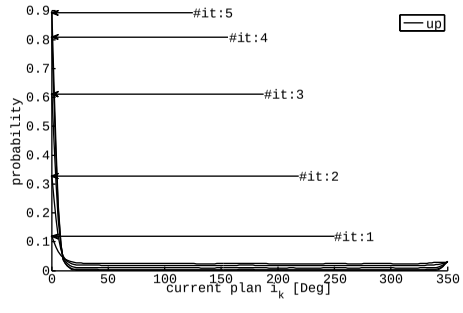
(a) front view user model



(b) top view user model

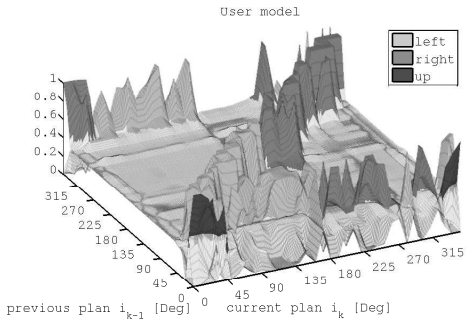


(c) posterior for action "left" and "right"

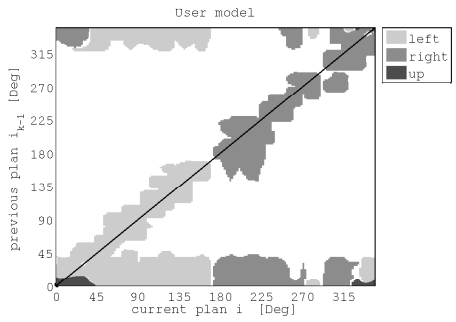


(d) posterior for action "forward"

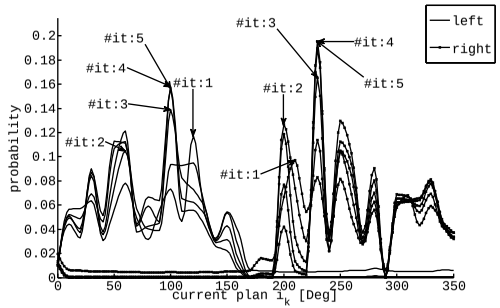
FIGURE 3.21: User model and posterior distributions of a discrete interface calibrated with the exp-distance-likelihood by looking only maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (0.2m, 0.1rad)$



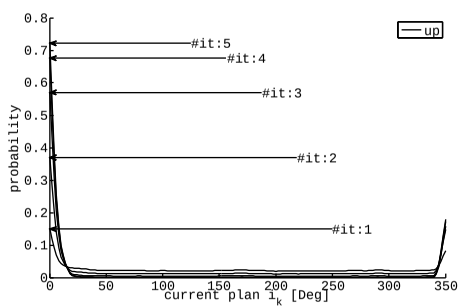
(a) front view user model



(b) top view user model

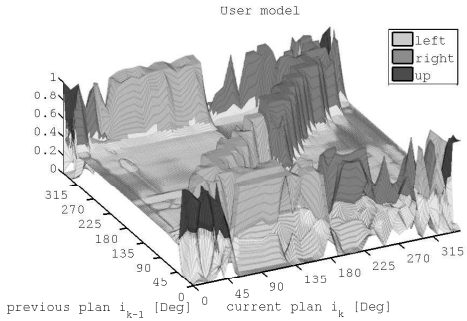


(c) posterior for action “left” and “right”

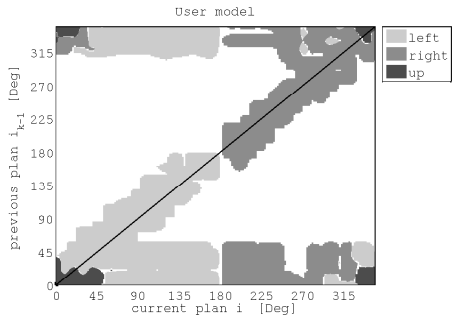


(d) posterior for action “forward”

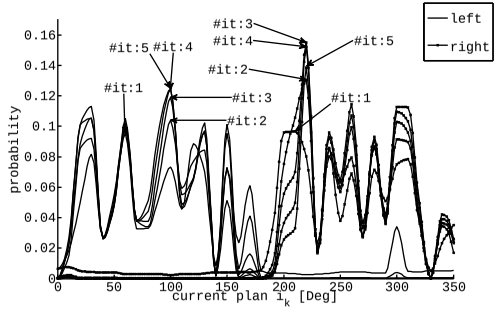
FIGURE 3.22: User model and posterior distributions of a discrete interface calibrated with the arctan-distance-likelihood by looking only maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (0.2m, 0.1rad)$



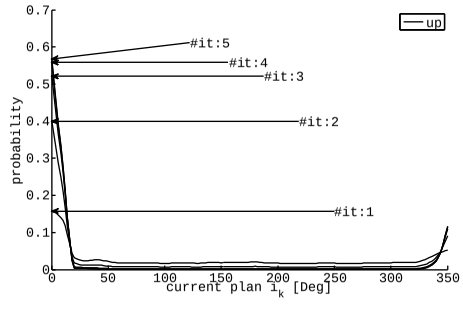
(a) front view user model



(b) top view user model



(c) posterior for action “left” and “right”



(d) posterior for action “forward”

FIGURE 3.23: User model and posterior distributions of a discrete interface calibrated with the cos-distance-likelihood by looking only maximal distance ahead, $\boldsymbol{a} = (\Delta l, \Delta\theta) = (0.2m, 0.1rad)$

3.3.2 Continuous interface

The previous section discussed the results of user modelling and plan recognition for a discrete interface. The *plan process probability* and *user model* can be represented as multinomial distributions in the discrete case, since actions $\boldsymbol{a}$ and user signals $\boldsymbol{u}$ are discrete and the number of user plans is finite. If user signals and robot actions are continuous, then the plan process probability can be calculated with the procedure of section 3.2.4 only if robot actions are quantised. We discuss user models for continuous joystick signals in section 4.3.

Let the robot's displacement between subsequent times $k-1$ and k be $\Delta\boldsymbol{x} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1} = [\Delta x \Delta y \Delta\theta]^T$. An estimate of the robot's velocity which moves the robot from pose $\boldsymbol{x}_{k-1}$ to pose $\boldsymbol{x}_k$ is:

$$\omega_k^{robot} = \frac{\Delta\theta}{\Delta T} \tag{3.34a}$$

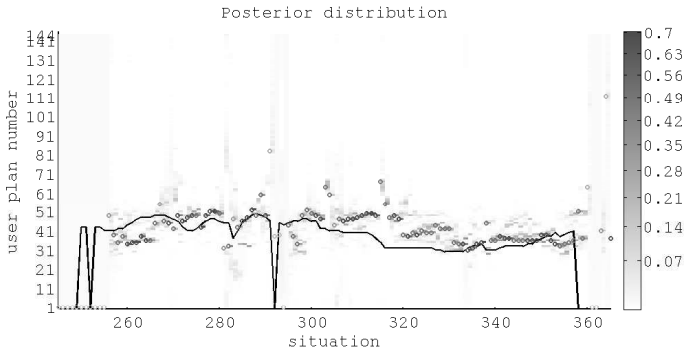
$$v_k^{robot} = \text{sign}\,(\Delta x_{rel}) \cdot \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\Delta T} \tag{3.34b}$$

$$x_{rel} = \Delta x \cdot \cos\theta_{k-1} + \Delta y \cdot \sin\theta_{k-1} \tag{3.34c}$$

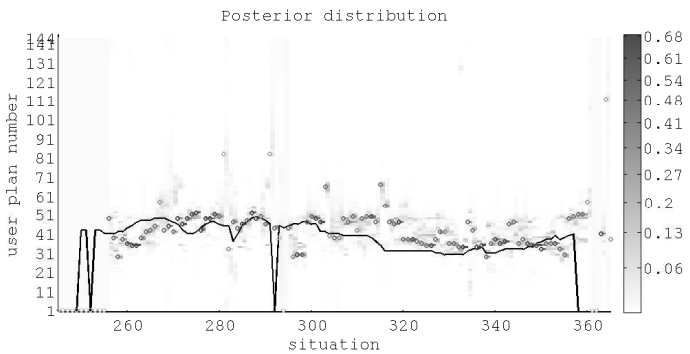
where x_{rel} is the x-coordinate of the next pose $\boldsymbol{x}_k$ expressed in the relative frame of $\boldsymbol{x}_{k-1}$. This relative displacement defines the sign of the velocity. The discrete robot action $\boldsymbol{a}_k$ equals the user plan most closely resembling the robot's velocity $\boldsymbol{v}_k^{robot} = (v_k^{robot}, \omega_k^{robot})$. In section 4.3.3 we discuss a velocity parametrisation of user plans, which makes such comparisons possible. We are aware that the Euclidean velocity norm is frame-dependent (Lin and Burdick, 2000), but if user plans and the robot's velocity are expressed in the same global reference frame, then comparisons are meaningful.

Figure 3.24 shows three posterior distribution over user plans. The solid line represents the plan most closely matching the robot's future path according to the metric discussed in section 3.2.4. The plan process probability of figure 3.24(a) is calibrated with the exp-distance likelihood, and for figures 3.24(b) and 3.24(c) with the arctan- and cos-likelihoods respectively. In all three cases, the posterior is calculated by combining the user model of a user diagnosed with spastic-tetraplegia with a prior over user plans. We estimate the user's plans for driving data that were not employed during calibration of either user model or plan process function. Section 4.3.4 contains a more detailed description of the experiments.

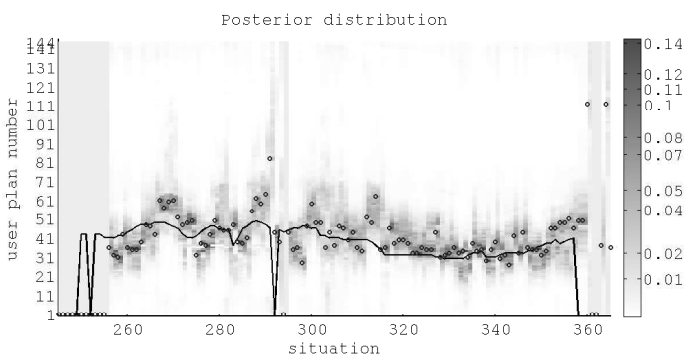
When comparing the posteriors for the different plan process functions, it is apparent that the exp- and arctan- distance likelihoods result in the most peaked posteriors, whereas the cos-likelihood posterior is spread wider around ground-truth. We believe that a higher level of noise is preferable to an overconfident posterior, which is occasionally wrong. Therefore, in section 4.3.4 we will discuss plan recognition using the cos-likelihood plan process function.



(a) exp-distance likelihood



(b) arctan-distance likelihood



(c) cos-distance likelihood

FIGURE 3.24: Posterior distribution of a continuous joystick user interface calibrated with three different distance-likelihoods.

3.4 Conclusions

This chapter defined a probabilistic framework for joint human-robot navigation. It described the estimation problem a robot faces when interacting with a human user. The robot is moving in a potentially unknown environment in which it needs to localise. Further, it is interacting with a user who does not communicate her plans explicitly.

An important contribution of this chapter is a formal specification of the complete state estimation the robot faces. We decompose this problem into two sub-problems in order to make the design of the navigation architecture more modular and in order facilitate the solution of each problem separately. The two sub-problems are Simultaneous Localisation and Mapping (SLAM) on the one hand, and a plan recognition on the other hand.

Concerning plan recognition, we have discussed the impact of global and local plans. Temporal reasoning with global plans is easy, but achieving a dense plan library is more involved as all homotopically different plans must belong to it. Conversely, local plans are beneficial for immediate navigation assistance, at the expense of increasing the complexity of temporal reasoning. Global references are lost for local plans, which would allow a straightforward initialisation of the prior for the next time step.

Therefore, the second major contribution in the chapter is a procedure to reason about the temporal evolution of local user plans. Our solution to local plan recognition relies on the insight that absolute certainty is only possible through global knowledge. In local plan recognition this global knowledge arises from the robot's trajectory, which is known during calibration. The robot's trajectory represents at each sampling time k the actual future, which we can compare to the potential futures of each user plan $\boldsymbol{i}_k$. We impose then that plans should inherit probability from their equivalent ancestors in terms of the robot's future trajectory.

Our solution of local plan recognition requires a measure to quantify the resemblance between trajectories. An important contribution of the chapter is therefore an analysis of candidate trajectory distances. Further, we studied the statistical properties of the chosen metric in order to understand the probabilistic structure of the problem. From this study we concluded that, besides the statistical properties of the trajectory metric, additional domain knowledge is needed to compute the plan process probability. This domain knowledge implies that a user plan is more likely to correspond to the robot's future trajectory the closer it lies to the robot's trajectory.

We formalised this domain knowledge in a subjective distance-likelihood, which we combined with other sources of uncertain knowledge to calibrate the plan process function. The rationale behind the distance-likelihood is to reduce the amount of data needed during training of the plan recognition models. Normally, it would be necessary to observe sufficient transitions between *all* possible user plans for all

possible actions. The method we proposed reduces this amount of data because knowledge about observed transitions propagates to unobserved plan transitions. Further, it also allows to control the level of noise of the plan process function to avoid overconfident posterior distributions.

The validity of the approach is demonstrated on a simulated wheelchair with a discrete interface, as well as for a real wheelchair with a continuous joystick. The experiments with the discrete input revealed the influence of the robot's kinematics on the plan recognition process. If the robot can turn over larger angles only, then plan recognition preserves this property.

In order to apply the same plan recognition procedure to a continuous interface it is necessary to discretise the actions of the robot. We proposed to assign robot actions to the user plan that would generate the robot's displacement. The experiments with continuous user inputs will be discussed further in section 4.3.4. The relevant insight of this chapter for the continuous input is that a higher level of noise in the plan process function is desirable over an overconfident posterior, which is occasionally wrong.

Chapter 4

User models for wheelchair driving

> *"[...] individuals carry their probabilities in their veins, and time, temptation, and circumstance will lead them at last to their fulfilment."*
>
> Javier Marías (⋆1951)
> *Your face tomorrow: Vol. II: Dance And Dream*, page 134 (Marías, 2008)

This chapter offers three perspectives of wheelchair navigation. The first view in section 4.1 characterises the potential user group of robotic wheelchairs. It describes the most common pathologies and injuries of the nervous system and how such pathologies affect wheelchair navigation.

Controlling a wheelchair is an instance of manual control, where the operator receives information through her senses and actuates mechanical devices in order to control the machine. Traditionally, the problem of manual control has been stated from a system-theoretic or control view. Therefore, our second perspective is system-theoretic and section 4.2 describes the relevant literature about manual control.

However, the control view is insufficient to describe wheelchair driving. During navigation, wheelchair users also perform high-level cognition in addition to compensating for perceived error signals. We bridge the divide between traditional manual control and high-level cognition in the third, *probabilistic*, perspective. This point of view is especially appealing in this context, since human performance is inherently non-repeatable, i.e. it is subject to stochastic noise. Furthermore, our models of wheelchair driving are uncertain because they necessarily simplify reality. Thus, the probabilistic view captures both modelling uncertainty and inherent

stochastic noise.

The goal of our probabilistic approach is to calculate the user signals' likelihood in function of a set of user plan hypotheses, as described in the previous chapter. This likelihood, which we call *user model*, allows to relate user inputs to a plan library in a way that sheds light on the user's current intention. Section 4.3 presents two user modelling approaches for continuous user input devices. First, in section 4.3.1, we introduce a first-principles linear regression user model. Starting from this model, we derive an analytical user model for a user with weak directional steering in section 4.3.2. Then, in section 4.3.3, we state a general learning technique, which can model the driving behaviour of arbitrary users. We illustrate the power of this general technique on a case study for a user diagnosed with spastic-quadriplegia in section 4.3.4. For this user, we derive a probabilistic user model of her driving and recognise her navigation plans.

The technical contributions of this chapter are chiefly in section 4.3 and the experimental analysis of section 4.3.4. The contribution concerning the first-principles linear regression user model is a mathematical justification of the model's validity. The interest for the analytical user model of a user with weak directional steering resides in its challenging requirements for shared control, as we will see in section 5.3. Further, in order to derive the analytical model, we introduce a principled way to measure driving performance by comparing steering disabilities to a reference user.

The most important contribution of the chapter is the general user model based on *Gaussian Process Regression* in section 4.3.3. We justify the selection of this technique with an analysis of user modelling requirements, and address how specific choices of the model fulfil them. In the experimental section we show how this model can extract a probabilistic driving model directly from data for a user with driving difficulties.

4.1 Potential user group of robotic wheelchairs

In this section we provide a list of the most common pathologies and injuries of the nervous system together with the symptoms that affect wheeled mobility. Most of the users characterised in this section cannot benefit from a standard power-wheelchair. They either lack the ability for fine motor control, or their cognitive and sensory limitations prevent them from navigating a power wheelchair safely. (Simpson et al., 2008) project a potential market of 1.4 to 2.1 million smart wheelchairs in the United States for users who could benefit from full or shared navigation assistance, at least part-time.

4.1.1 Pathologies and injuries of the nervous system

Alzheimer Disease (AD) is now the most common neuro-degenerative disease, with more than 20 million cases worldwide. It results in abnormal protein deposits in the brain, where tau and Amyloid-β proteins assemble into *amyloid*-like filaments (Goedert and Spillantini, 2006).

In 1997, the prevalence of Alzheimer's disease in the United States was 2.32 million (Brookmeyer et al., 1998); of these individuals, 68% were female and 32% male, presumably due to the higher life expectancy of women. The incidence rate of persons living with Alzheimer's disease increases with age, from 4.3% for 75-, 8.5% for 80-, 16% for 85- and 28.5 for 90-year old (Brookmeyer et al., 1998). Currently, the annual number of new incident cases is 360,000 (Brookmeyer et al., 1998).

Cerebral Palsy (CP) is a disorder resulting from an injury to the brain before its development completed. The injury can be caused either before, during, or after birth since the brain continues to develop until the second year of life. Cerebral Palsy is a non-progressive condition. It is characterised by motor and, in about two third of the cases, intellectual dysfunction.

Around 70% to 80% of patients with CP present spastic clinical features. This involves deep tendon reflexes, tremors, muscular *hypertonicity*, weakness and scissoring[1]. 10% to 20% of patients are affected by *athetoid* or *dyskinetic* CP. *Athetoid* CP implies slow writhing movements of some or all limbs, especially during periods of stress. *Ataxic* CP afflicts approximately 5% to 10% of patients and results in balance and coordination impairments. Intentional tremor is another symptom present in *ataxic* CP, which interferes with fine motor control.

Neurological dysfunctions common to all types of Cerebral Palsy include communication difficulties, impaired vision or hearing, and abnormal touch or pain perception (Krigger, 2006).

In 2001, the prevalence of Cerebral Palsy in the United States was of 764,000 cases. An estimated 8000 babies and infants, plus 1200 to 1500 preschool-age children are diagnosed with Cerebral Palsy every year in the United States (Krigger, 2006).

Cerebrovascular Accident (CVA) or stroke is caused either by deficient supply of blood to the brain (*ischaemic* stroke), or by copious discharge of blood from blood vessels (*haemorrhagic* stroke) (Donnan et al., 2008).

The most common origin of *haemorrhagic* stroke is the hypertensive small-vessel disease. Blood vessels rupture after dilating due to an abnormal amount of blood. Around two-thirds of primary cerebral *haemorrhage* patients have been previously diagnosed with hypertension. The remaining patient group generally suffers some

[1] Closing of the legs as the tips of a scissor would close.

form of intracranial vascular malformations or cerebral *amyloid angiopathy*. In this case, cerebral blood vessels are obstructed by a translucent substance consisting of proteins (Donnan et al., 2008).

About 80% of all strokes are *ischaemic*. In such type of strokes, the inflow of arterial blood to the brain is obstructed, which leads to unrecoverable damage to parts of the brain. Within the *ischaemic penumbra*, the functionally impaired but structurally intact region surrounding the *ischaemic* core, a cascade of neuro-chemical events is triggered. Ultimately, those events lead to cell death, but the process can be halted through various neuro-protective therapies. The *ischaemic* core represents the irreversible stage of the degenerative cascade of events (Donnan et al., 2008).

Depending on which part of the brain is affected, CVA can result in physical, cognitive, and visual impairments of various degrees. Symptoms interfering conventional wheelchair use include visual field neglect, visual field loss, spasticity and *hemiplegia*[2].

Cerebrovascular Accident is the cause of 9% of deaths around the world and after *ischaemic* heart disease, stroke is the second most common cause. Stroke incidence rate ranges between 240 per 100, 000 to 600 per 100, 000, depending on the region of the globe. An estimated 500 people per 100, 000 survive a stroke, but have to live with its consequences (Donnan et al., 2008).

Severe Traumatic Brain Injury is a brain damage resulting from external forces. Typical examples include direct impacts, rapid accelerations and decelerations, blasts, or penetrating objects. The nature, duration, distribution, and intensity of the forces determine the type of injury.

The injury manifests macroscopically in focal contusions, *haematomas*, diffuse swelling, and shearing of white-matter tracts. On a cellular level, the damage appears as micro-poration of membranes, leaky ion conduction, and changes in proteins. Also, blood vessels can be torn and *micro-haemorrhages* be present. Traumatic brain injury might cause diffuse *axonal* injury involving scattered small lesions of the white-matter tract. Frequently, widespread *ischaemic* brain damage can appear in addition to the primary damage (Maas et al., 2008).

The severity of brain injuries is measured with the Glasgow Comma Scale. This scale assesses eye, motor, and verbal capacity of the patient in order to determine the degree of injury. Depending on which part of the brain is damaged, severe brain injury can result in physical, cognitive, and perceptual impairments. Potential symptoms include visual field neglect, *hemiplegia*, tremors, *bradykinesia*, fatigue, and post-traumatic epileptic seizures. Impaired attention, impulse control and deficits in executive reasoning can originate from the injury as well (Simpson et al., 2008).

[2]Total or partial paralysis of one side of the body.

In the United States, the prevalence of brain injury is of 5.3 million people with an incidence of 1.5 million people per year. Out of this population, approximately 10% are diagnosed with severe brain injury (Simpson et al., 2008).

Spinal Cord Injury is an injury of the nervous system due to mechanical trauma. Fractured or displaced bone fragments, disc material, or ligaments can damage the central and peripheral nervous systems. This occurs through rupturing of blood vessels, disruption of axons and breaking of neural-cell membranes. A few minutes after the trauma, the spinal cord swells until occupying the entire spinal canal at injury level. When the swelling exceeds the body's blood pressure, auto-regulation of blood flow ceases and the body is affected by systemic *hypotension*. *Hypotension* reinforces the deficient blood supply (*ischaemia*) even further. The combined effect of *ischaemia* and the release of toxic substances from perforated neural membranes triggers a biochemical reaction which damages healthy neighbouring neural cells (McDonald and Sadowsky, 2002).

The estimated prevalence in the United States is $183,000$ to $230,000$ cases, with a yearly incidence of $10,000$ new cases. Injuries result primarily from motor vehicle accidents $(36-48\%)$, violence $(5-29\%)$, falls $(17-21\%)$, and recreational activities $(7-16\%)$. Average age at injury is 31.7 years with a male to female ratio of four in one (McDonald and Sadowsky, 2002). Around 23% of cases involve injuries above the fourth cervical vertebra (C4), which leads to complete tetraplegia.

Amyotrophic Lateral Sclerosis (ALS) is a progressive, neuro-degenerative, and fatal disease. Motor neurons in the lateral part of the spinal cord and the brain stem are gradually scarred (*sclerosis*). As a consequence, muscle waste away progressively (*atrophy*) in a period ranging from months to few years. Other common symptoms of ALS are weakness, uncoordinated muscular twitching involving simultaneous contraction of contiguous muscle fibres (*fasciculations*), and increased reflexes. ALS rarely affects cognitive functions. Gradually, patients become completely disabled requiring in the terminal stages ventilatory support and gastronomy. Within approximately five to ten years of the initial diagnosis patients decease (Walling, 1999).

In 1999 ALS affected approximately $30,000$ people in the United States with an annual incidence rate of one to two cases per $100,000$. Amyotrophic Later Sclerosis affects more frequently men than women and is typically diagnosed in middle age. The cause of the disease is yet unknown and there exists no cure for it, but existing drugs can alleviate its symptoms. (Walling, 1999).

Multiple Sclerosis (MS) is an inflammatory autoimmune disease of the central nervous system. *Sclerosis* refers to the scars or lesions which appear on the axons after periods of inflammatory *de-myelination*. The isolating *myelin* sheath covers nerve axons and provides electrical isolation as displayed in figure 4.1. This *myelin*

sheath gets damaged by the body's own immune system in attacks to the *myelin* producing *oligodendrocyte* cells (Compston and Coles, 2002). The disease evolves in episodes with recovery and also in episodes leaving persistent deficits. Other possible evolutions are a continuous disease progression or a combination of relapses and progressive degeneration (Compston and Coles, 2002).

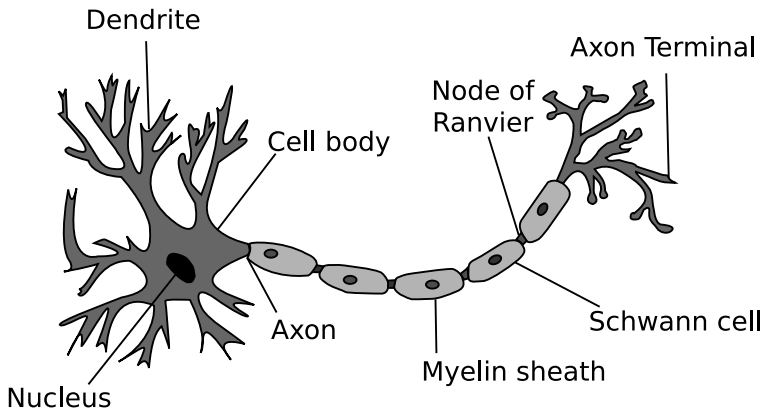


FIGURE 4.1: Structure of a neuron (Wiki Commons: Q. Jarosz, 2010).

The presence of certain symptoms is difficult to predict and varies from patient to patient, depending on which part of the central nervous system is affected. Cognitive impairments such as deficits in attention, in reasoning, and in executive function can appear early on in the evolution of the disease. Dementia occurs mostly in latter stages if the lesions are located in the cerebrum. Lesions in the optical nerve, if present, lead to unilateral painful loss of vision, obscuration of part of the visual field, and reduced visual acuity. Intentional tremors result from lesions in the cerebellum and cerebellar pathways. The consequences of damage in the brainstem are clumsiness and poor balance, difficulties in limb coordination, and gait *ataxia*. Vision is also impaired in case of brainstem lesions. Common symptoms of vision impairments are involuntary, rapid, oscillating movement of the eyeballs (*nystagmus*), and diplopia, which means that objects are seen double. Plaques in the spinal cord cause weakness, stiffness and painful spasms (Compston and Coles, 2002).

Multiple sclerosis affects twice as many women as men. The disease has an incidence of about 7 per $100,000$ every year, prevalence of around 120 per $100,000$ (Compston and Coles, 2002).

Parkinson's Disease is a neuro-degenerative disorder related to ageing. The disease originates when *dopaminergic* neurons degenerate in the *substantia nigra pars compacta*, a part of the brain dedicated to fine motor control and reward-conditioned learning. Another pathological feature of Parkinson's Disease are

intracytoplasmic inclusions known as *Lewy bodies*. Degeneration of nerve cells and *Lewy bodies* can also be present in the *locus ceruleus*, *nucleus basalis*, *hypothalamus*, *cerebral cortex*, cranial nerve motor nuclei, and central and peripheral components of the autonomic nervous system (Olanow and Tatton, 1999).

Parkinson's disease causes resting tremors, rigidity, *bradykinesia* or slowness, gait disturbance, and postural instability. Currently, Parkinson's Disease is treated with the *dopamine* precursor *levadopa*, which can cross the blood-brain barrier since *dopamine* itself cannot. However, $5-10$ years after treatment onset, many patients develop adverse effects such as *dyskinesia* and motor response fluctuations. Furthermore, *levadopa* therapy cannot halt disease progression associated with development of dementia, autonomic dysfunction and postural instability (Olanow and Tatton, 1999).

Parkinson's Disease affects 0.5% of individuals aged 60 to 69 years and 2.5% of people older than 80. The prevalence of the disease in the United States is of $894,000$ individuals with $8,000$ to $18,000$ new cases diagnosed each year (Simpson et al., 2008).

Multiple System Atrophy (MSA) belongs to the group of neuro-degenerative diseases and is frequently mistaken for Parkinson's Disease due to its similar set of symptoms. Pathologically, it is characterised by cell loss and *glial cytoplasmic* inclusions in several central nervous system structures. Another feature of MSA is *gliosis*, which means an excessive development of *glia*. *Glia* cells are non-neuronal cells that maintain *homeostasis*, form *myelin*, and provide support and protection for the brain's neurons, spinal cord, and *ganglia* (Wenninga et al., 2004).

Clinically, two types of multiple system atrophy are distinguished, the Parkinsonian type, MSA-P, and the *ataxic* type, MSA-C, with a 80% to 20% distribution respectively. Despite both clinical presentations having similar survival times, in patients with MSA-P the functional deterioration is more rapid (Wenninga et al., 2004). Symptoms of MSA-P include progressive *akinesia* and rigidity, jerky postural tremor, and less commonly, resting tremor. *Dystonia*, difficulty in articulating words, and postural instability are also prevalent among MSA-P patients. MSA-C symptoms comprise gait *ataxia*, limb kinetic *ataxia*, and *oculomotor* disturbances.

The prevalence is estimated to range from 1.9 to 4.9 cases per $100,000$ people, and a yearly incidence rate of 0.6 per $100,000$ people. MSA affects men and women equally and disease onset generally occurs within the sixth decade of life with a mean survival time of $6-9$ years (Wenninga et al., 2004).

Progressive Supranuclear Palsy is the second most common Parkinsonian neuro-degenerative disorder, frequently mistaken for Parkinson's Disease. It involves neuro-degeneration, *gliosis*, and abnormal accumulation of tau protein in the *basal ganglia*, brain stem, *prefrontal cortex*, and *cerebellum*. Tau protein stabilises the neuron's *cytoskeleton* and is always present in healthy individuals. In Progressive

Supranuclear Palsy, however, the tau protein partially crystallises forming unusual deposits of tangled fibres called *neurofibrillary tangles*. These fibres destroy affected neurons in a chain of cellular events involving *mitochondrial* dysfunction and oxidative stress (Zampieri and Fabio, 2006).

Symptoms of Progressive Supranuclear Palsy comprise early postural instability with repeated falls, vertical gaze, palsy, speech and swallowing problems, *bradykinesia*, axial rigidity, and sub-cortical dementia. Walking is clumsy, slow, and unsteady. Generally, after an average of 5 years, patients cannot walk independently and require the use of a wheelchair. Vision impairments also appear early on in the disease, including slowness of vertical *saccades*, difficulty in opening or closing the eyes voluntarily, spasms of the eyelid's muscle, and decreased blinking frequency (Zampieri and Fabio, 2006).

The prevalence for Progressive Supranuclear Palsy has been estimated in 1.39 cases per 100,000 people with a yearly incidence rate of 5.3 per 100,000 people aged above 50 years. The disease onset is in average between 60 and 65 years, affecting equally men and women. Survival time normally is around 7 years, with *pneumonia* as the typical cause of death (Zampieri and Fabio, 2006).

4.1.2 Symptoms affecting wheeled mobility

This section describes different types of impairments, which affect the usage of powered wheelchairs. The estimates for the number of people suffering each impairment reflect the values of various studies cited in (Simpson et al., 2008).

Physical impairments of the upper body

An important factor determining whether a patient is prescribed a manual or powered wheelchair is usually the presence of impairments that reduce the physical strength or coordination of the upper body, and in particular of the arms. Often, these kinds of impairments interfere as well in the use of powered wheelchairs. Upper body impairments might prevent quick reactions to unforeseen obstacles, from driving long distances, or from completing tasks that require extensive hand-eye coordination.

Ataxia is a lack of muscular coordination resulting in a jerky, unsteady, oscillating motion of the trunk or the limbs. Normally, opposing muscle groups ensure smooth and graceful movement. Contraction and relaxation of opposing muscles is coordinated by a portion of the brain called the *cerebellum*. Diseases that damage the *cerebellum*, spinal cord, and peripheral nerves connecting the *cerebellum* to the muscle groups can interfere with fine muscular control. As a result movements are coarse, jerky, or uncoordinated (Kantor, 2007).

Ataxia is common to diagnoses of Multiple Sclerosis (23% – 84% of cases) and Cerebral Palsy (5% – 10%) (Simpson et al., 2008). People suffering *ataxia* can have difficulties with wheelchair navigation, when fine motor control or fast reactions are required.

Bradykinesia means a reduced speed or amplitude of movement. It can interfere with one's ability of reacting to dynamic obstacles. Diagnoses involving *Bradykinesia* include Multiple System Atrophy (71% of cases), Progressive Supranuclear Palsy (22%), severe Traumatic Brain Injury (26%) and Parkinson Disease (12.5%) (Simpson et al., 2008).

Spasticity is a set of symptoms causing stiff or rigid muscles due to continuous contractions. This condition can interfere with walking, movement, or speech. Spasticity symptoms include scissoring (crossing of the legs as the tips of scissors would close), repetitive jerky motions (*clonus*), unusual posturing, and carrying the shoulder, arm, wrist, and finger at abnormal angles. This condition generally results from damage to the part of the brain that controls voluntary movement. It may also occur when the nerves travelling from the brain down to the spinal cord have been damaged (Kantor, 2007). Diagnoses that can produce spasticity include Cerebrovascular Accident (35% – 51% of cases), Cerebral Palsy (70% – 90%), Multiple System Atrophy (10%) Multiple Sclerosis (65% – 90%), and Spinal Cord Injury (12% – 37%) (Simpson et al., 2008).

Dystonia is characterised by abnormal postures or movements caused by involuntary muscle contractions. Abnormal movement is more pronounced in the presence of emotional stress. However, it may disappear during sleep (Kantor, 2007). *Dystonia* can be problematic if fine motor control is required for steering a wheelchair. Possible diagnoses where *dystonia* appears include Multiple System Atrophy (43% of cases) and Cerebral Palsy (17.6%) (Simpson et al., 2008).

Fatigue or weakness. Users experiencing muscular fatigue or weakness might collide with suddenly appearing obstacles if they cannot react in time. Travelling long distances might also be problematic. Diagnoses, which include muscle fatigue or weakness are Multiple Sclerosis (43% – 90% of cases), severe Traumatic Brain Injury (37% – 50%) and Amyotrophic Lateral Sclerosis (26%) (Simpson et al., 2008).

Tremor is a rhythmic contraction of one or more muscles. Tremors can occur when the affected limb is resting (resting tremor). Intentional tremors appear during execution of fine motor tasks. Generally, users affected by resting tremors can navigate a powered wheelchair. On the contrary, intentional tremors might interfere with fine navigation tasks or cause collisions with obstacles. Diagnoses,

where tremors are present include Multiple Sclerosis (75% of cases), Cerebral Palsy (10% – 20%), Multiple System Atrophy (67%), Progressive Supranuclear Palsy (12% – 16%), and severe Traumatic Brain Injury (26%) (Simpson et al., 2008).

Cognitive impairments

In order to navigate a wheelchair safely, cognitive skills such as navigation, path planning, and problem solving are required. A wheelchair user should also be able of controlling her emotions and impulses, given that a wheelchair can become potentially dangerous to its user and other people in the surroundings. When cognitive impairments are present, it might be difficult to plan trajectories to distant locations or to stay concentrated on the executed path. Furthermore, cognitive impairments might interfere with perceiving dangerous situations such as suddenly appearing obstacles.

Impaired attention, agitation, or impulse control. A neurological impairment can affect concentration, attention and mood control. This might lead a wheelchair user to become confused, distracted or agitated, and, in the worst case, cause aggressive or violent behaviour. Diagnoses where these impairments might appear include Alzheimer Disease (48% of cases), and severe Traumatic Brain Injury (23% – 60%) (Simpson et al., 2008).

Impaired executive reasoning refers to deficits of cognitive skills involved in goal-directed behaviour. To achieve her goals, the user should be able to judge different situations, reason about them, plan a strategy, solve problems, take decisions, and sequence her actions according to a pre-selected plan. Some or all of these processes are impaired in diagnoses such as Cerebral Palsy (64% of cases), Progressive Supranuclear Palsy (50%), Multiple System Atrophy (0.5% – 17%), Multiple Sclerosis (30%–70%), severe Traumatic Brain Injury (55%), and Alzheimer Disease (41%) (Simpson et al., 2008).

Visual Impairments

Visual impairments are especially problematic for wheelchair navigation due to deficient perception of obstacles and navigational cues. Navigation in narrow corridors, or through doorways, where fine hand-coordination is required, might also present difficulties.

Low vision or blindness. Operating a conventional wheelchair with almost no vision poses a considerable challenge. Using guide-dogs or canes to aid wheelchair navigation is seldomly seen in practice.

Impaired head or neck movement leads to a reduced visual field even without suffering from low vision or blindness. If the head or neck cannot be moved, the driver is at risk of colliding with obstacles or door post. Most notably, when driving backwards, obstacles in the back of the wheelchair cannot be perceived. Diagnoses where impaired head or neck movement is symptomatic include Spinal Cord Injury above the C4 level (100% of cases), Amyotrophic Lateral Sclerosis (26%), and Multiple Sclerosis (43% − 90%) (Simpson et al., 2008).

Impaired eye movement can result from *nystagmus* (rapid movement of the eyeballs from side to side), difficulty initiating *saccades*, and poor *saccadic* pursuit. These symptoms are present in Multiple System Atrophy (restricted down gaze in 10% of cases, *nystagmus* in 23% of cases) and Progressive Supranuclear Palsy (6% of cases) (Simpson et al., 2008).

Visual field loss reduces the perception of the environment. For wheelchair users, this translates into an increased risk of collision with obstacles and a difficulty to perceive navigation cues. Loss of visual field can be present in Cerebrovascular Accident (20% of cases) (Simpson et al., 2008).

Visual field neglect implies that an individual ignores the part of the world she cannot perceive. As a consequence, reaction times to objects in the neglected visual field are much slower and the neglected space is less frequently scanned for obstacles. Diagnoses involving visual field neglect are right hemisphere Cerebrovascular Accident (13% to 82% of cases), left hemisphere Cerebrovascular Accident (0% to 76%) and severe Traumatic Brain Injury (45%) (Simpson et al., 2008).

4.2 Manual control

In manual control a human is receiving information about a machine's state through her senses (visual, tactile, and vestibular) and tries to actuate on the state of that system through mechanical devices such as handles, knobs, and pedals, or any other input mechanism. The goal is to minimise error or a more complex function of the inputs. The human operator forms thus an integral part of the human-machine system, since she is effectively closing the control loop. Therefore, the global performance of the human-machine application cannot be derived from the characteristics of either part in isolation.

Human-machine performance depends on a set of *task variables*, which define the manual control task. Users show different performance depending on the characteristics of the reference input they are tracking or they are trying to anticipate. Repetitive inputs are for instance easier to track than random appearing signals. Disturbances, whether intrinsic or extrinsic, can also have a significant

impact on the task at hand. If disturbances are within the operator's bandwidth, then the operator will try to compensate for them in order to improve her overall performance. This limited bandwidth imposes constraints on the type of process dynamics that humans are able to control. In section 4.2.1 we describe subjective ratings of different plant dynamics regarding their controllability. Other task variables concern the human-machine interface. The type of information display triggers different control behaviours, as will become clear in the ensuing paragraphs. The other part of the human-machine interface is the input device that controls the system. Ergonomic or haptic interfaces can improve a system's controllability.

Environmental conditions also influence manual control tasks. The operator's workload by additional tasks generally reduces her performance. Environmental factors include ambient illumination, acoustic noise, and temperature.

The third class of task parameters are *operator-centred variables*. With training it is possible to improve performance, as long as it is possible to extract patterns from the task. Different operators might also present various levels of skill. Experienced pilots usually outperform novices if they can transfer their skills to the particular manual control task. A very important factor for good task performance is the operator's motivation. Highly motivated operators can excel at difficult tasks by maintaining a high state of arousal and concentration. Normally, it is hard to maintain such levels of concentration for extended periods of time. The amount of fatigue might thus reduce task performance, which is the last operator-centred parameter.

A given experiment is described in terms of *procedural variables*. Operators might perform differently depending on the type of instructions received. For instance, they can react to anticipated situations as opposed to completely unexpected events. Further, they might have to fulfil a strictly constrained task or be allowed more freedom in selecting the appropriate strategy. As mentioned in the operator-centred variables, training improves performance. Thus, operators who are allowed to practice prior to the experiment will probably perform better than if they have to learn the task along the way.

This classification is according to (Sheridan and Ferrel, 1981).

Manual control behaviour. As outlined previously, an important task variable concerns the presentation of inputs because it can lead to different control behaviours. Figure 4.2 displays the four types of manual control humans typically encounter.

Depending on the display of inputs, broadly defined as the quantities to which an operator reacts, humans show *compensatory*, *pursuit*, *preview* or *precognitive* control (Sheridan and Ferrel, 1981).

In a *compensatory* situation the human only perceives the error, $e(t)$, defined as the difference between an ideal reference input, $r(t)$, and the output of the plant, $y(t)$.

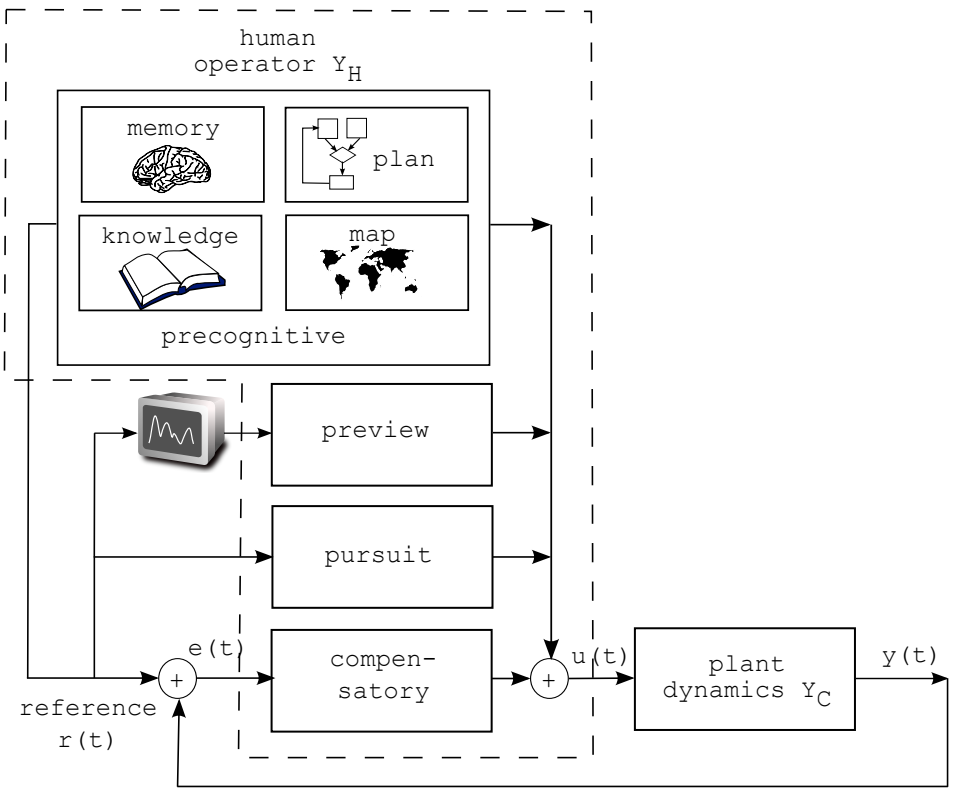


FIGURE 4.2: Classification of manual control situations as *compensatory*, *pursuit*, *preview* or *precognitive* control.

Most models in manual control literature are for compensatory control (Sheridan and Ferrel, 1981).

Pursuit control means that the operator acts on the system based both on the error as well as on the instantaneous reference input, $r(t)$. Therefore, the human can exert a certain amount of feed-forward control. For random input in simple tasks there is no significant difference in performance between compensatory and pursuit systems. However, for difficult input signals or plants requiring significant operator lead, pursuit displays are superior, since the operator can make predictions about the input from its direct observation (Sheridan and Ferrel, 1981).

A *preview system* extents a *pursuit system* by displaying a limited time forecast of the reference input. The operator can anticipate future variations and react accordingly. Normally, the preview-time equals the interval which allows an appropriate reaction to changes (Sheridan and Ferrel, 1981).

In *precognitive control* the operator has prior knowledge of the reference input in

different terms than through direct observation or preview. The foreknowledge can be about low-level properties of the reference, such as its statistical distribution, its amplitude, or its temporal and frequency variability. At a higher level, *precognitive control* also implies that the human has a certain plan to achieve a goal and sequences her actions accordingly. Hence, the operator can generate her own reference input according to her strategy and goals. Two cognitive processes are of special importance during generation of this reference, namely memory and reasoning. The operator can resort to pre-trained control strategies stored in memory for each possible contingency, or devise new strategies by reasoning about the tasks. Memory and reasoning mostly occur simultaneously, since devising a new plan can also benefit from previously encountered situations. External guiding elements such as maps, instructions, or signs also aid cognition to select a certain control strategy (Sheridan and Ferrel, 1981).

Research on manual control models dates from the time posterior to World War II. Significant progress in the field was achieved during the 1950s until the late 1970s for *compensatory tracking* applications. Two kinds of models received most attention in the manual control literature and are still widely accepted nowadays. The first is a quasi-linear model, where the human operator is represented in the frequency domain by a transfer function plus a certain amount of uncorrelated remnant. The second type of models propose an optimal control representation of the human operator, who is trying to optimise certain task-dependent performance criteria.

4.2.1 Quasi-linear model of the human operator

The quasi-linear model proposed by (McRuer and Krendel, 1957) describes human output in manual control compensatory tasks[3]. In the quasi-linear model, the operator displays a linear transfer function acting on the observed error plus an uncorrelated remnant. Experimentally, the following transfer function was calibrated for human performance in compensatory control:

$$U(j\omega) = Y_H(j\omega) \cdot E(j\omega) + \Phi_{remnant} \tag{4.1a}$$

$$Y_H(j\omega) = \frac{K \exp(-j\omega\tau_d)(T_L j\omega + 1)}{(T_N j\omega + 1)(T_I j\omega + 1)} \tag{4.1b}$$

where $U(j\omega)$ is the control output of the human, $E(j\omega)$ is the error signal in figure 4.2 and $\Phi_{remnant}$ an uncorrelated remnant. T_N represents a first-order-lag coefficient of approximately 0.2 seconds determined by the time to initiate muscle displacement. τ_d is the reaction-time delay ranging between 0.12 and 0.2 seconds. K, T_L and T_I are a gain, lead and lag coefficients, which can be adjusted by the human depending on the task. The model is quasi-linear since the transfer function

[3]The operator just observes the error defined as the difference of reference input and controlled system output.

parameters vary for each combination of reference inputs and controlled element dynamics.

The crossover model

The experimental data reported by (McRuer et al., 1965) for controlled elements K_C, $K_C/j\omega$, $K_C/(j\omega-2)$,$K_C/(j\omega)^2$, $K_C/j\omega(j\omega-1/T)$ could be fitted to different settings of control gain and operators by the model stated in equation 4.1, or by extensions thereof, even if some of the controlled elements were unstable.

The major finding of their work is that for different reference inputs and controlled element dynamics, the combined open-loop response of human and controlled system, $Y_H Y_C(j\omega)$, approximately equals in the region of the crossover frequency:

$$Y_H Y_C(j\omega) = \frac{\omega_c \exp(-j\omega\tau_\epsilon)}{j\omega}, \text{ near } \omega_c \tag{4.2}$$

where τ_ϵ is a delay and ω_c is the crossover frequency. At ω_c the gain of the open-loop combined human and controlled system response equals unity, $|Y_H Y_C(\omega_c)| = 1$. This model implies that the operator equalises her lead or lag terms in order to reach an open-loop slope of -20 db/dec in the region of the crossover frequency. Outside of that range, the crossover model is both inaccurate and inappropriate. However, its predictions are essentially correct because the dominant closed-loop dynamics are rarely influenced by the shape of the open-loop transfer function in regions far from the crossover frequency (McRuer and Weir, 1969).

The parameters ω_c and τ_ϵ are task dependent. For low-pass forcing functions (reference signal $r(t)$ in figure 4.2), the crossover frequency ω_c is linearly dependent of the forcing function bandwidth ω_{ref}. Further, the crossover frequency ω_c has a constant offset linked to the controlled element dynamics, $\omega_{c0}(Y_C(j\omega))$.

$$\omega_c = \omega_{c0}(Y_C(j\omega)) + \Delta\omega(\omega_{ref}), \qquad \Delta\omega(0) = 0 \tag{4.3a}$$

$$\tau_\epsilon = \tau_0(Y_C(j\omega)) - \Delta\tau(\omega_{ref}), \qquad \Delta\tau(0) = 0 \tag{4.3b}$$

$$\tau_0(Y_C(j\omega)) \cdot \omega_{c0}(Y_C(j\omega)) = \frac{\pi}{2} \tag{4.3c}$$

$$\phi_{margin} = \measuredangle(Y_H Y_C(\omega_c)) - (-\pi) = \frac{\pi}{2} - \tau_\epsilon \cdot \omega_c \tag{4.3d}$$

The time delay, τ_ϵ, summarises operator reaction times, neuromuscular dynamics, as well as any high frequency lag of the controlled element. It depends linearly on the forcing function bandwidth and has a constant offset due to $Y_C(j\omega)$. Contrary to the crossover frequency, the operator's delay τ_ϵ decreases with input signal bandwidth (McRuer and Jex, 1967). The values of τ_ϵ and ω_c are selected by the human operator to minimise rms error.

Remnant and model validity

Inter-subject variation in the crossover region is smaller than at higher or lower frequencies. This suggest a uniform control strategy for the critical stability region.

The quasi-linear model considers only the linearly correlated part between controlling output and reference signal. Therefore, its validity depends on the ratio of human response power linearly correlated with the forcing function divided by the total response power, ρ^2. If ρ^2 strongly deviates from unity, the model fails to predict manual control behaviour. Experimental evidence shows that ρ^2 is low for high gains K_C of the controlled element as well as for systems that are difficult to control. Remnant increases also with forcing function bandwidth and the order of the differential equation the human must apply for achieving the crossover model condition, *i.e.* $Y_H Y_C = K/j\omega$.

According to the experimental measurements, the remnant at different frequencies has a smooth spectrum without peaks. Non-linearities in human control strategy can be ruled out since no harmonics of the reference input were observed (McRuer et al., 1965). (McRuer et al., 1965) attribute remnant primarily to non-stationary parameters. (Sheridan and Ferrel, 1981) also explain it as humans injecting random noise into the system in the neighbourhoods of the forcing frequencies.

The crossover model is valid for randomly appearing inputs in compensatory tracking applications for bandwidths lower than 10 rad/s. These include many practical manual control applications suited for humans. The crossover model fails for highly non-linear systems, systems with many unstable poles, or systems with an order higher than two. However, humans are unsuited for these applications as well.

Subjective ratings

For a controlled system with response $Y_C = K_C/j\omega(Tj\omega + 1)$, the pilot has to adopt a lead term $T_L j\omega + 1$ to compensate T. Human pilots perceive the system more difficult to control the higher the lead. Pilots also dislike lag-terms in mid to high frequency range. For the unstable plant $Y_C = K_C/j\omega(j\omega - 1/T)$ pilots had to decrease their delay τ_ϵ in order to compensate for unstable roots. This was perceived as less comfortable due the high neuromuscular tension which is necessary to achieve such behaviour. Optimal control comfort is furthermore achieved at moderate gain values (McRuer and Jex, 1967).

Crossover model in automobile driving applications

(Weir and McRuer, 1970) have applied the basic crossover model to compensatory control of automobiles with random appearing inputs such as unanticipated road bends and random disturbances such as wind gust. Compensatory control is

relevant for this application because the driver perceives a guidance cue in her visual field and, as a response to disturbances, steers the vehicle to follow the cue and reduce tracking error in a compensatory fashion.

(Weir and McRuer, 1970) compare possible feedback loops of different input quantities on the basis of the closed-loop performance a driver is able to achieve through equalisation. Heading-angle and heading-rate offer the best results in terms of bandwidth and controllability for single-loop manual control, as compared to inertial lateral deviation or lateral velocity. However, minimising lateral deviations from the middle lane or any reference is essential for good path-following behaviour. Control based on lateral error requires significant lead equalisation by the driver, which is perceived as uncomfortable. Therefore, (Weir and McRuer, 1970) propose multi-loop feedback control representation of the human operator, where the inner-loop controls either heading or heading-rate, and the outer-loop acts on the lateral deviation from the middle of the lane.

Similarly, (Hess and Modjtahedzadeh, 1990) propose a driver model for lane-keeping on a curved roadway based on a four variable state-space. The relevant variables are lateral velocity, yaw- or heading-rate, heading angle, and lateral deviation. The model differentiates between high- and low-frequency compensation. The high-frequency compensation corresponds to an inner feedback loop closed through the driver's own proprioceptive muscular system. The outer loop performs a low-frequency compensation with a single lead term that can be adapted to the driving task. The model meets bandwidth and stability requirements even for aggressive steering tasks. Further, it considers the neuromuscular system and verifies the crossover model conditions. Three fundamental properties achieve the model's good control performance. Firstly, the open-loop transfer function exhibits high gains in the low frequency range. Secondly, around the crossover frequency the open-loop transfer function exhibits a slope of $-20\ dB/dec$, as required by the crossover model. Thirdly, the model attenuates higher frequencies considerably.

4.2.2 Optimal control model

The assumption of optimal control models is that a well-trained and motivated operator performs near optimally, subject to her psychophysical constraints and her degree of task understanding. If the human is familiar with her own dynamics, with the dynamics of the controlled process, as well as with possible stochastic variability in her performance, it is reasonable to suppose that she might behave optimally. The optimal operator possibly tries to trade-off control effort, error-response and time-to-completion.

Optimal control models state a certain optimality criterion with a number of free parameters, which can be adjusted to observed data. The models measure the degree of deviation from a given optimal behaviour. The problem remains, though, of finding the optimality criterion that humans employ for each task.

(Kleinman et al., 1971) propose an optimal model of the human for controlling a linear dynamical system with random disturbances. The operator monitors the controlled process from a display and estimates its past dynamics from a delayed and noisy observation of its state. Further, the operator anticipates the system's current state with a time-delay predictor, as humans generally tend to compensate for their intrinsic delays. Based on the current state's prediction, the human produces optimal motor commands according to the following criterion:

$$J(\boldsymbol{u}) = \frac{1}{T}\int_0^T (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \dot{\boldsymbol{u}}^T\boldsymbol{R}\dot{\boldsymbol{u}})dt \tag{4.4}$$

where the integrand is a quadratic function with $\boldsymbol{Q}$ and $\boldsymbol{R}$ positive definite.

Including the control output change rate, $\dot{\boldsymbol{u}}$, in the cost instead of the output, $\boldsymbol{u}$, penalises rapid control movements, which are rarely exerted by trained pilots. It also reflects more accurately the way a human operates when the neuromuscular lag parameter, T_N, is large. An adapted representation of the optimal control model after (Kleinman et al., 1971) appears in figure 4.3.

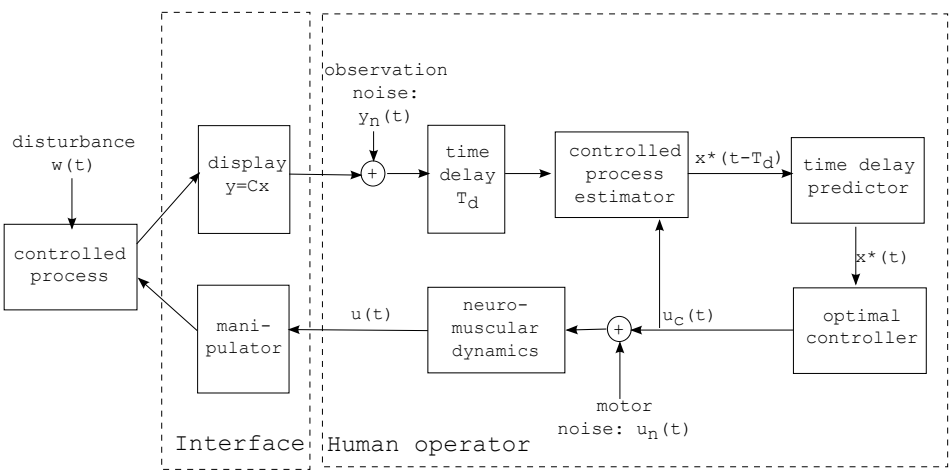


FIGURE 4.3: Optimal control model for a human operator after (Kleinman et al., 1971).

The model quantizes human psychophysical limitations in four parameters. Humans react with a certain delay to variations of the task. This is represented by the aggregated perceptual time delay at the input (T_d). Further, observation ($\boldsymbol{y}_n$) and motor ($\boldsymbol{u}_n$) noises corrupt the inputs and desired outputs respectively. These noises model stochastic variability in manual control. They comprise random perturbations of the human response-function, variability of model parameters, and random errors in observing control system outputs and producing system

inputs. The control finally exerted on the controlled system depends on the first-order neuro-motor dynamics, which transform the operator's desired output, $\boldsymbol{u}_c$, corrupted by noise, $\boldsymbol{u}_n$, into the control signal $\boldsymbol{u}(t)$:

$$T_N \dot{\boldsymbol{u}}(t) + \boldsymbol{u}(t) = \boldsymbol{u}_c(t) + \boldsymbol{u}_n(t) \tag{4.5}$$

where $\boldsymbol{u}(t)$ is the control output exerted on the manipulator, $\dot{\boldsymbol{u}}(t)$ the control output change rate, T_N the lag parameter of the neuromuscular system, $\boldsymbol{u}_c$ the optimal controller output and $\boldsymbol{u}_n$ the motor noise. Values for the parameters reported in (Kleinman et al., 1971) are a delay of $T_d = 0.15\ s - 0.25\ s$ and a neuromuscular lag of $T_N = 0.1\ s - 0.3\ s$, with $T_N = 0.1\ s$ being typical. The observation noise is of variance 0.001π times the observed signal's variance ($var(\boldsymbol{y}_n) = 0.001\pi \cdot var(\boldsymbol{y})$), whereas the motor noise is of variance 0.003π times the optimal controller output's variance ($var(\boldsymbol{u}_n) = 0.003\pi \cdot var(\boldsymbol{u}_c)$).

(Kleinman et al., 1971) validate the model on simple compensatory tracking tasks for controlled element dynamics K_C, $K_C/(j\omega)$, and $K_C/(j\omega)^2$. Experiments on the more complex task of controlling the longitudinal position of a hovering aircraft in a Vertical Take-Off and Landing (VTOL) flight simulator are reported as well.

Drawbacks of optimal-control models The main drawback of optimal-control models of human performance lies in choosing the optimisation criterion. During the experiments there is generally no access to the subject's internal optimisation criteria. Moreover, optimality can be achieved with different criteria each governed by a distinct set of model parameters.

4.2.3 Adaptation to changing task requirements

Initial adaptation or learning of a task can be achieved in a variety of ways. The first option is to force the operator's limbs through the intended motion. However, retention of such learning is relatively poor. The second option is to demonstrate a certain task and to give immediate feedback on the operator's performance. An even better way of retaining a task is to let the operator experiment through trial-and-error and to give a reward once she discovers the correct response (Sheridan and Ferrel, 1981). An example of the latter type of learning is when drivers inject deliberate step-responses on the brakes and steering wheel to check the car's behaviour when it suddenly drives over ice or patches of water.

Once the operator has learned a task, one of her most prominent features is her adaptability to changes. In order for a modification in behaviour to occur, the operator has to detect first a variation in the task. There might be some time delay if the operator does not anticipate any modifications. Prior to adapting, humans

identify the type of change in order to optimise performance with respect to a new set of task requirements.

As pointed out by (Young, 1969), humans can adapt to changes of a controlled process at four levels. Firstly, they can adapt to the input. Secondly, adaptation to the controlled plant is also possible. Thirdly, an operator can adapt to the task if she modifies her strategy in function of penalties to errors. Further, depending on the time-to-completion or other optimality criteria, the operator might adapt to the task even when neither control element nor reference input have changed. Fourthly , the operator can select a pre-programmed adaptation strategy depending on environmental conditions. For instance, the condition of the road surface or changing weather patterns can trigger different pre-programmed behaviours.

In manual control the first two types of adaptability have been extensively studied and will be explained in the next two sections.

Input adaptation

Input adaptation refers to the capacity of detecting patterns in the input as well as to the ability of adapting to properties of the input such as its bandwidth.

It has been observed that humans actuate open-loop on the controlled element if they detect patterns in the sensed input. These patterns generate precognitive feedforward responses, when applied to an internal estimate of the inverse plant model. Therefore, the findings reported previously for manual compensatory tasks apply only to random appearing inputs. Signals with periodic components are detected and their variation anticipated. This effect increases the operator's effective bandwidth, especially at low frequencies, where humans are able to compensate for plant dynamics. Above 1 Hz, the produced output patterns might loose synchronisation with the input. For this reason, operators normally resort to a combined strategy of precognitive control and tracking in order to correct prediction errors. The weighting of compensatory tracking and precognitive control varies with time and depends on the input predictability, measured by the success of estimating the input correctly (Young, 1969).

Adaptations of the crossover model to the input's signal bandwidth have already been discussed in section 4.2.1.

Controlled element adaptation

Plant dynamics can vary at three time scales. On one end of the spectrum are fast changes in gain or dynamics, due to e.g. unexpected system failures. A well-trained operator adapts to sudden modifications by discrete mode-switches. On the other extreme are slowly varying plant dynamics, which can be modelled as quasi-stationary sequences of time-invariant dynamics. Changes on an intermediate

frequency range between $0.1 - 1\ Hz$ directly interfere with the pilot's control actions, and are thus difficult to detect and to model.

In compensatory tracking tasks, humans adapt rapidly to changes in the controlled process' gain. Adaptation to small gain changes are unconscious. On the contrary, large variations including sign reversals are readily detected because of a significant increase in the error. In (Young, 1969) laboratory experiments of several authors are reported, where well-trained operators reacted to expected but unknown changes in control gain. Polarity reversals of the plant caused first an exponential divergence for about $0.5\ s$, to which the subjects accommodated by inverting their own control magnitude. If the gain was increased, test subjects could track again the input after a series of oscillations. For a combination of gain increase and signal reversal, operators tried first to correct polarity in order to avoid instabilities and subsequently adjusted the gain. The subjects tended to overcompensate for large errors after about $0.4 - 0.8\ s$ by overshooting in the opposite direction. Reduction of the error to its stationary level occurred after around $1 - 3\ s$.

More extended experiments studied how, besides changes in polarity and gain, well-trained pilots reacted to modifications of the plant transfer function. Simultaneous increase of gain, change of polarity and different plant transfer function resulted in longer adjustment times. The polarity inversion could be detected and corrected after approximately $1\ s$. Damped oscillatory periods of around $20\ s$ followed the correction of polarity, since the subjects had to adapt not only to a new gain, but also to a completely different transfer function. Several parameters of the crossover model had to be equalized to the new dynamics. Auditive cues reduced the detection time, but left the identification time unaffected.

In practice, adaptation times are expected to be larger than the reported values since changes are not anticipated and might be more difficult to detect. For the operator to achieve the lowest adaptation times, a thorough training on both the actual and the changed dynamics is recommendable. Early detection can also be improved by a pursuit display, where the operator can rapidly identify a change in response.

4.2.4 Reaction times

Definition of reaction time. The reaction time measures the time elapsed between presentation of a stimulus and a detectable response. This includes physiological signal processing times as well as information processing times. An external signal must first activate the appropriate stimulus receptor, which relays afferent signals to the brain. In the brain different sensor modalities might be integrated and interpreted. Then, there is normally some delay to select a suitable action, which has to be transmitted back to the muscles. Muscles contract as a response to the efferent signals arriving through the peripheral nervous system (Sheridan and Ferrel, 1981). Reflex responses are generally faster since part of the cognitive processing is bypassed.

Already since the 19*th*-century the factors having an impact on reaction times have been documented. Reaction time is typically increased for unexpected stimuli, for multitude of stimulus-response pairs, and for unfamiliar stimuli. If it is difficult to discriminate among stimuli, if a stimulus requires searching or scanning, if the stimulus-response relation is complex, or if the response is difficult to execute, then reaction times tend to be slower as well. Different sensor modalities also lead to different reaction times. Visual cues take longer to process than auditory stimuli, which in turn are slower processed than tactile stimuli (Sheridan and Ferrel, 1981).

Reaction times for breaking and steering a car

Reaction times have been widely studied for a huge variety of tasks under very different conditions. Most related to the problem of driving is the body of literature dealing with braking and steering reaction times. An interesting comparative study of braking reaction times with a short reference to steering is provided by (Green, 2000). The author identifies braking reaction time with mental processing and movement times. Mental processing is further subdivided into sensation time, perception time and response selection and programming times.

Information-processing during braking. (Green, 2000) describes human operation within two information-processing modes, *automatic* response and *attentive*- or *controlled*-mode. Automatic response refers to situations where there is a strong innate stimulus-response compatibility such as when avoiding obstacles on collision course, and to situations requiring highly trained responses to common signals. Automatic responses are fast because they lack conscious decision making and the response resembles more a reflex. Hitting the brake pedal as response to flashing braking lights of the preceding car is an example of a trained automated response.

Conversely, *attentive* or *controlled mode* occurs as a reaction to novel events which require more processing time for perception and decision-making. For instance, a driver reacting to an unexpected object appearing suddenly on the road in conditions of poor visibility would perform attentive processing. Normally, attention ranges in a continuum from automatic to attentive mode. Reaction times in situations corresponding to automatic processing are generally faster than for attentive processing.

Expectancy. The variable having the highest impact on braking response times reported by (Green, 2000) is *expectancy*. Green subdivides studies of response times into those where the driver is reacting to an expected event from those where the braking situation is completely unexpected. The first case serves as a measure for a lower bound of reaction time under ideal circumstances, whereas the second group tries to quantify reaction time under more realistic conditions. For expected event studies, the driver is instructed to brake as response to a signal with low temporal and spatial uncertainty, such as stopping after a fixed trajectory or in response to

auditory or visual cues. Several studies coincide on a bound of around $0.70s - 0.75\ s$ for the ideal braking reaction time, where $0.5s - 0.55\ s$ are perception time and $0.2\ s$ movement time. These values correspond to very favourable conditions, such as low uncertainty, intense visual, foveally viewed signal, but no practice.

A further cluster of studies reported in (Green, 2000) refers to common situation but which are uncertain on their onset. Those situations include braking as a response to flashing of braking lights, or the response to changing of a traffic light. Even if these situations are expected during normal driving, the driver cannot anticipate them exactly. Under these circumstances reaction times vary between $1.2\ s$ and $1.35\ s$ with standard deviations of around $0.6\ s$.

The studies cited by (Green, 2000) of truly unexpected events describe reaction times for intrusions of objects into the driver's path. Although these events happen occasionally during driving, their probability is very low and average drivers lack a trained response. Response times in all stages of mental processing are slower. Detection of the object is slower since it first appears within the peripheral visual field. Perception is slower as well because unusual events require more time to be interpreted correctly. Additionally, the driver must choose between braking or steering, which leads to a higher response selection time. Movement times average $0.3\ s$, a 50% increase compared to the ideal movement time after an expected event. Under realistic driving conditions, the overall reaction time ranges from $1.5\ s$ to $1.8\ s$ depending on the time-to-collision. Steering is about $0.3\ s$ faster since no foot movement from throttle to braking pedal is required.

Age and gender. The impact of age or gender on braking reaction time is inconclusive according to (Green, 2000). Although one might expect slower reaction times for older people, experience and fitness of active aged drivers offsets their slower reactions. Furthermore, ageing affects people differently. Older drivers form a very heterogeneous group and the reported test data are noisier. Some research concludes that the effect of ageing on braking reaction times is more pronounced at higher cognitive load, for instance, while talking on the mobile phone. However, the exact magnitude of the ageing effect is unknown. The effect is expected to be higher in poor visibility conditions, given that vision might be worse for old people. Even if braking times seems to increase with age, older drivers compensate it by generally driving more slowly.

Urgency, measured in terms of time-to-collision, decreases the braking reaction time since drivers are in greater state of arousal. However, when time-to-collision is very short, braking reaction time paradoxically increases. One study reported in (Green, 2000) found that braking reaction time increased $0.4\ s$ when time-to-collision diminished from $4.35\ s$ to $2.85\ s$. When braking alone is inappropriate, the driver must consider alternative evasive actions such as steering into the oncoming lane, or braking and steering. This introduces additional time for response selection,

and potentially for response planning since the driver might have to check for traffic in the oncoming lane.

Cognitive load. (Green, 2000) also cites studies showing that braking reaction time increases due to a higher cognitive load. For instance, when the lead car varies speed and the road has more turns, the reaction time increases. Use of visual in-car displays also has a negative effect on braking time because of the increasing visual processing time for the dual visual task. Impact of mobile phones on brake reaction times is controversial. It is expected that visual tasks are much less impaired by a concurrent auditory task than by a secondary visual activity. Some studies report an increase in reaction time from 0.4 s to 0.5 s. However, other research reports a decrease in reaction time. This decrease in breaking time might stem from the driver's higher state of attention while using a mobile phone. The driver might compensate for her risky behaviour by paying more attention to the road. In general, drivers tend to respond slowly during high cognitive load, but the effect is not easy to quantify. When signals are subtle, such as acceleration or deceleration of the lead car, the effect on braking time is longer than half a second. Meaningful signals that break through the attentional filter, e.g. braking or traffic lights, have an effect of around a few tenths of a second at most (Green, 2000).

4.3 User models for wheelchair driving

The purpose of this section is to provide a probabilistic view of wheelchair control. *Compensatory* manual control, which we describe in section 4.2, is insufficient to model wheelchair driving. Wheelchair users are not only responding to perceived momentary tracking errors originated by random disturbances. Instead, wheelchair driving relies heavily on *precognitive* and *preview* control. Wheelchair users combine prior knowledge about the system with previous navigation experience in order to plan trajectories to desired destinations. They can anticipate how the system reacts and thus can program their actions accordingly. Additionally, they can observe the desired reference trajectory, even if they are unable to execute it perfectly. Furthermore, wheelchair users normally do not track a wheelchair riding in front of them, nor have to maintain the course inside a lane.

However, parallels to compensatory manual control exist as our user plan representation considers the path to follow. Our framework generates 'virtual lanes' the user is tracking in order to understand the cognitive process of navigation. This modelling effort is influenced by two sources of uncertainty. Firstly, human performance is inherently stochastic. Even the most trained driver cannot repeat exactly the same manoeuvre in similar situations. The driver's reasoning about navigation tasks might also be imperfect, which results in a certain level of noise. Secondly, any model simplifies reality. Our modelling capabilities of wheelchair driving are consequently subject to uncertainty. For this reason, we believe that a

probabilistic model can offer the most complete view of the type of manual control required for wheelchair navigation.

The appeal of our probabilistic framework to model wheelchair navigation is not only theoretical. It has a practical component that allows a robotic wheelchair to reason about the plan of its user and provide *customised assistance*. In order to provide this assistance, the robot needs to estimate the posterior probability distribution over user plans based on the user's current input, available free space, and past driving behaviour. Please refer to chapter 3 for a more detailed description of the *plan recognition* process. The focus of this section is on the personalised component of the framework that allows to adapt *plan recognition* to a specific user.

In order to clarify the probabilistic view on wheelchair navigation we first introduce an example. Let us assume that the user's current configuration is as shown in the left side of figure 4.4. Calculating the probability of docking at the table (estimating the posterior) directly is more difficult than obtaining the probability of the input signals $\boldsymbol{u}_k$ that are necessary to reach the table, assuming the user has a plan $\boldsymbol{i}_k$ to reach the table in mind. The latter probability is generally lower dimensional than a probability over continuous user plans.

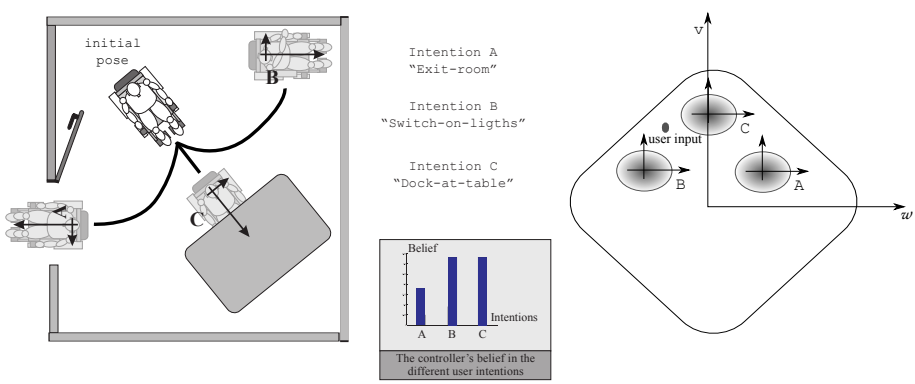


FIGURE 4.4: The left side of the figure shows the wheelchair in its present configuration along with three potential destinations. Bayes' theorem allows to calculate the posterior probability of each destination, given the input of the user and past driving behaviour. This is achieved through a user model, which provides the likelihood of the present input under the assumption the user has a certain plan in mind. The user model is represented on the right part of the figure, where each possible trajectory results in a probability distribution in user input space. Darker colours indicate the region where user inputs are more likely for a given intention.

Formally, we define the probability of the user's current input $\boldsymbol{u}_k$ given plan $\boldsymbol{i}_k$ as the *user model*, $p_{user}\left(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \mathcal{H}_{1:k}\right)$. The right hand side of figure 4.4 shows such

a user model, which is a probability distribution in user-input space rather than in user-plan space. Reasoning about user plans given user inputs has been inverted through Bayes' rule into a problem of reasoning about user inputs given user plans.

Definition 4.1 *The user model* $p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \mathcal{H}_{1:k})$ *in this work expresses the likelihood of current user inputs* $\boldsymbol{u}_k$*, given that the user has plan* $\boldsymbol{i}_k$ *in mind, and given a prior history,* $\mathcal{H}_{1:k}$*, of user inputs (*$\boldsymbol{u}$*), robot actions (*$\boldsymbol{a}$*), and constant parameters (*$\boldsymbol{h}$*) such as maps:* $\mathcal{H}_{1:k} = \{\boldsymbol{u}_{1:k-1}, \boldsymbol{a}_{1:k}, \boldsymbol{h}\}$.

This section discusses the user model p_{user}, which is the cornerstone of our Bayesian plan recognition framework. The user model allows to adapt the plan recognition process to each user. $p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \mathcal{H}_{1:k})$ is user-specific, since it encapsulates how a particular user transforms mental plans into inputs to the robot. The accuracy and usefulness of the framework depends to a large degree on the predictive power of the employed model. The presence of steering disabilities introduces additional complexity for estimating the likelihood of the user's input, since not all trajectories can be executed directly by the user.

The structure of the remainder of this chapter is as follows. First, in section 4.3.1, we describe a first-principles linear user model for able-bodied users based on (Demeester et al., 2006; Demeester et al., 2008). Then, in section 4.3.2 we report a user model that is manually tailored to the driving abilities of a disabled person, which we presented in (Huntemann et al., 2007a). We applied the first principles user model and the manually-tailored user model to a set of global trajectories during *plan recognition*. However, these models are equally valid for local trajectories.

The last section on user modelling, section 4.3.3, proposes a general machine learning technique suitable for *any* kind of user, and which avoids manual adjusting of its parameters. Furthermore, this model is also compatible with a local plan library. Local plan recognition, although beneficial for local navigation assistance, introduces additional complexity regarding the combination of present and past plan estimates. Therefore, as explained in section 3.2, a plan process function is needed to transfer probability across time. We explain the procedure to acquire a local plan process function in section 3.2.4, and the necessary adjustments for a continuous user input device in section 3.3.2.

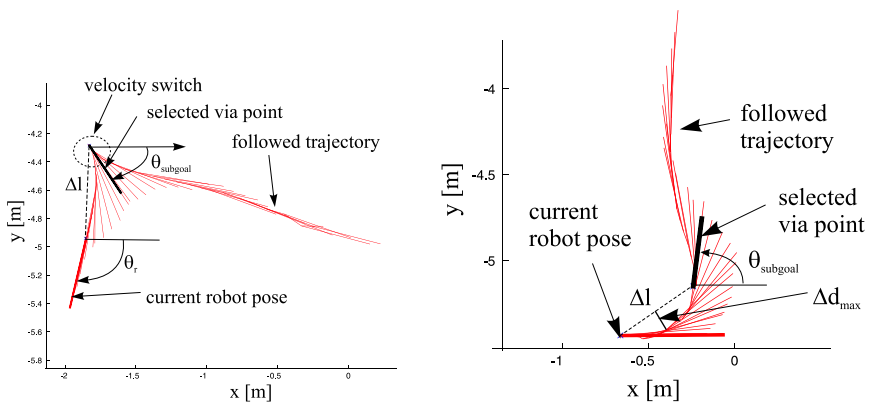
In the experimental section, section 4.3.4, we model the driving pattern of a user diagnosed with spastic-quadriplegia. These results extend the results in (Huntemann et al., 2008a) by increasing the number of plan hypotheses, by comparing different parameterisations of user plans, and by performing the full probabilistic plan recognition for local trajectories. The work reported in (Huntemann et al., 2008a) deals mainly with user modelling.

4.3.1 First-principles linear regression user model

The first-principles linear user model is only outlined here. More information and experimental results can be found in (Demeester et al., 2006; Demeester et al., 2008).

In the ideal case, where the user can execute any plan, we hypothesize that the user steers a wheelchair by tracking perceptual cues on a mental path. Path tracking errors are specified in terms of differences between the current robot pose and these subgoals. We estimate then the parameters of the tracking controller through least-squares regression.

We propose to place the subgoal at the position where a velocity switch occurs, i.e. where the linear velocity changes its sign (cf. figure 4.5(a)). If no velocity switch occurs, a certain point $\boldsymbol{p}_{via}$ on the trajectory plays the role of subgoal. This subgoal verifies that the distance to the trajectory of any point on the segment linking the subgoal $\boldsymbol{p}_{via}$ to the robot's current pose $\boldsymbol{x}_k$ is at most Δd_{max} (see figure 4.5(b)).



(a) Trajectory with velocity switch. (b) Trajectory without velocity switch.

FIGURE 4.5: Position of via points on possible trajectories.

From these subgoals, we determine the regressors $\Delta\theta$ and Δl_{cor} as follows:

$$\Delta\theta = \theta_{subgoal} - \theta_r + \omega_r \cdot \widehat{T}_{sys} \tag{4.6a}$$

$$\Delta l_{cor} = \Delta l \cdot \left(\frac{1 + \cos(\Delta\theta)}{2} \right)^{\alpha} + v_r \cdot \widehat{T}_{sys}. \tag{4.6b}$$

where $\boldsymbol{t_r}(v_r, \omega_r)$ is the robot's twist, $\widehat{T}_{sys}$ is an estimate of the system delay computed as the peak of the correlation between user input and robot speeds and

Δl is the distance to the subgoal. This distance Δl is weighted with a bell-shaped function parametrised by α in equation 4.6b, which models the observed behaviour that wheelchair drivers first tend to turn until they are more or less aligned with the via point before they drive forward. θ_r is the robot's current orientation.

The regressors $\Delta\theta$ and Δl_{cor} can predict the linear and rotational velocity inputs of the user, $\boldsymbol{u}_k^M = \left(v_k^M, \omega_k^M\right)$, to track plan $\boldsymbol{i}_k^M$, with the following regression model:

$$v_k^M = b_1 \cdot \Delta l_{cor}\left(\boldsymbol{i}_k^M\right) + \varepsilon_v \ ; \qquad \varepsilon_v \sim \mathcal{N}\left(0; \sigma_v\right) \tag{4.7a}$$

$$\omega_k^M = a_1 \cdot \Delta\theta\left(\boldsymbol{i}_k^M\right) + \varepsilon_\omega \ ; \qquad \varepsilon_\omega \sim \mathcal{N}\left(0; \sigma_\omega\right) \tag{4.7b}$$

where σ_v and σ_ω are constant noise levels estimated from the regression's variance.

The likelihood of a given user signal $\boldsymbol{u}_k^{ref} = (v_u, \omega_u)$ is then calculated from the predicted user signal $\boldsymbol{u}_k^M = \left(v_k^M, \omega_k^M\right)$ as follows:

$$p_{user}\left(\boldsymbol{u}_k^{ref} | \boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) \propto \exp\left(-\frac{\left(v_u - v_k^M\right)^2}{2\sigma_v^2}\right) \cdot \exp\left(-\frac{\left(\omega_u - \omega_k^M\right)^2}{2\sigma_\omega^2}\right) \tag{4.8}$$

Properties of the model. We provide in the following a mathematical intuition for the properties of the first-principles linear model.

We claim that for low turning speeds and via-points near the current robot pose, linear and angular controls are independent and linearly linked to the angular, $\Delta\theta$, or linear ΔL, displacements respectively.

Assuming the user plans can be locally approximated by a circumference of arc, then the motion equations are:

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \frac{v}{\omega}\left(\sin(\theta + \omega\Delta t_{via}) - \sin(\theta)\right) \\ \frac{v}{\omega}\left(\cos(\theta) - \cos(\theta + \omega\Delta t_{via})\right) \\ \omega\Delta t_{via} \end{bmatrix} \tag{4.9}$$

where $\boldsymbol{a} = (v, \omega)$ are the controls which generate the approximated mental path of the user. From motion equation 4.9 we can compute the angular, $\Delta\theta_{via}$ and linear,

ΔL_{via}, displacements.

$$\Delta L_{via}^2 = \Delta x^2 + \Delta y^2 = 2\frac{v^2}{\omega^2}\left(1 - \cos(\omega \Delta t_{via})\right) \tag{4.10a}$$

$$\Rightarrow \Delta L_{via} = \pm 2\frac{v}{\omega}\sqrt{\frac{1 - \cos(\omega \Delta t_{via})}{2}} = 2\frac{v}{\omega}\sin\left(\frac{\omega \Delta t_{via}}{2}\right) \tag{4.10b}$$

$$\Rightarrow \Delta L_{via} \approx 2\frac{v}{\omega} \cdot \frac{\omega \Delta t_{via}}{2} = v \Delta t_{via} \tag{4.10c}$$

$$\Delta \theta_{via} = \omega \Delta t_{via} \tag{4.10d}$$

Equations 4.10 require a via-point in the proximity of the robot's current pose and a low angular speed $|\omega|$. Then the controls (v, ω) necessary to track that via-point are a linear function of the linear and angular displacements respectively.

A global wave-front planner (Demeester et al., 2005), which yields more complex trajectories than arc circumferences, generated the user plans in (Demeester et al., 2006; Demeester et al., 2008). Therefore, the previous intuition might not be completely accurate. However, for an able-bodied user and a set of 32 potential user plans, this first-principles user model could predict certain manoeuvres to reach a set of global intentions.

However, the via-point hypothesis and linearity requirements impose serious restrictions on the type of driving styles that can be modelled. The first-principles linear user model assumes an ideal user, perfectly capable of tracking any path by narrowly focussing on close navigation cues.

4.3.2 Analytical user model for users with weak directional steering capabilities

Estimating the user's input likelihood, i.e. the user model, becomes more complex in the presence of driving difficulties. The user might fail to execute a trajectory she has in mind, which somehow has to be reflected in the user model.

A possible solution proposed in (Huntemann et al., 2007a) is to split the user model p_{user} in two components, a *reference user model* and a *performance model*. The *reference user model* represents the generic stochastic variability in wheelchair driving without reference to a specific user. The performance model, on the other hand, correlates a user-specific steering disability with the idealised driving of a reference user. By separating the generic stochastic variability from the user-specific component, it is possible to monitor the user's performance through a quantitative model of her driving. This performance is always related to an idealised user. Hence, it becomes possible to detect if the user's driving is worsening or improving in time.

Mathematically, splitting the user model into a reference and a performance model corresponds to marginalizing out the joint probability distribution of the ideal reference signals, $\boldsymbol{u}_k^{ref}$, and the signals of a disabled user, $\boldsymbol{u}_k^{dis}$.

$$p_{user}\left(\boldsymbol{u}_k^{dis}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) = \int p\left(\boldsymbol{u}_k^{ref}, \boldsymbol{u}_k^{dis}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) d\boldsymbol{u}_k^{ref}$$

$$= \int p_{perf}\left(\boldsymbol{u}_k^{dis}|\boldsymbol{u}_k^{ref}\right) \cdot p_{user}\left(\boldsymbol{u}_k^{ref}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) d\boldsymbol{u}_k^{ref} \tag{4.11}$$

where:

1. $p_{perf}\left(\boldsymbol{u}_k^{dis}|\boldsymbol{u}_k^{ref}\right)$ is the performance model relating the distorted user input in presence of steering disabilities to the reference user model. It is assumed independent of previous history $\boldsymbol{\mathcal{H}}_{0:k}$ and only dependent on the current reference signal.
2. $p_{user}\left(\boldsymbol{u}_k^{ref}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right)$ is the reference model of an able-bodied user.

Performance model for driving disabilities

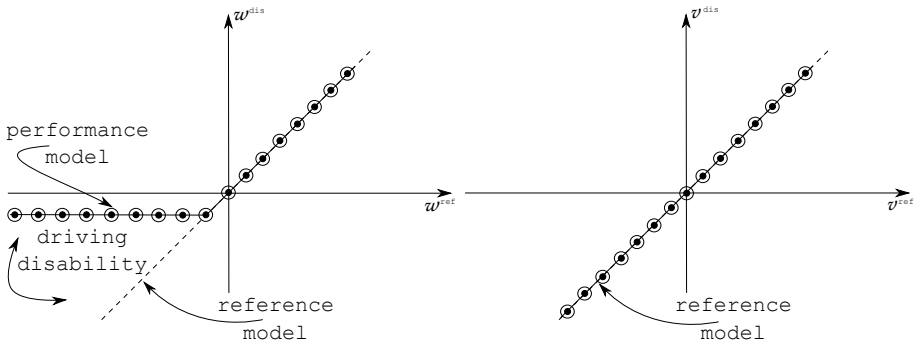
Any form of driving disability can be modelled in relation to a reference user. Such relationship does not have to be time-invariant, which allows to model varying performance. Figure 4.6 shows an example of such a functional relation between a reference user and a user whose left turning signal is weak.

In the wide sense, all deviations from the straight line with unit slope in the performance model can be classified as driving disabilities. Such broad definition allows to model effects like fatigue for able-bodied users as well as any kind of driving behaviour of disabled users. The performance model can be regarded as the adaptive component in the framework for plan recognition and shared control. It has to be adapted to each particular user and her steering performance.

Derivation of the user model. For the disability depicted in figure 4.6 it is possible to compute the stochastic component in the performance model analytically. Above a certain threshold, this user lacks the strength to issue left-turning commands.

$$\omega^{dis} = \begin{cases} \omega_{th} & \omega^{ref} \geq \omega_{th} \\ \omega^{ref} & \omega^{ref} < \omega_{th} \end{cases} \tag{4.12a}$$

$$v^{dis} = v^{ref}, \forall v^{ref} \tag{4.12b}$$



(a) Angular velocity (b) Linear velocity

FIGURE 4.6: Example of a personalized performance model for a user with a weak left signal. The angular speed is in hardware specific units (hardware units NV: In SI units turning left implies $\omega(SI) > 0\ rad/s$ and in NV's $\omega(NV) < 0\ NV$). The performance model relates signals of users with driving disabilities to a reference user. Any deviation from the dashed line can be classified as driving disabilities no matter if it arises from actual disabilities or from temporary effects such as fatigue. The circles on top of the line represent Dirac delta functions. Other probability distributions are conceivable, at the expense of an analytically intractable integral in equation 4.11.

where $\omega_{th} \geq 0\ rad/s$ is the threshold value corresponding to the inflection in the performance model of figure 4.6. Although some users might present such symptoms, the interest of this particular driving limitation resides in its challenging requirements. Considerable information is removed from the user's input. Therefore, we will be able to demonstrate the benefits our framework for plan recognition and shared control for this driving disability, if the user is able to reach poses in the environment (read intentions) lying at her left when she intents to do so.

In order to calculate the performance model, the linear and angular velocities are assumed conditionally independent:

$$p_{perf}\left(\boldsymbol{u}_k^{dis}|\boldsymbol{u}_k^{ref}\right) = p_{perf}\left(\omega^{dis}|\omega^{ref}\right) \cdot p_{perf}\left(v^{dis}|v^{ref}\right) \tag{4.13}$$

Determining the performance model for the linear velocity is straightforward, since the linear velocity is not affected by the driving disability. Thus, $p_{perf}\left(v^{dis}|v^{ref}\right)$ is a Dirac delta function centred on the reference linear velocity. Introducing this performance model in equation 4.11 yields a change in variables after marginalizing out the reference linear velocity by the properties of Dirac's delta function.

The performance model for the angular velocity can be determined with Bayes' rule as explained in the following equation:

$$p_{perf}\left(\omega^{dis}|\omega^{ref}\right) = p\left(\omega^{ref}|\omega^{dis}\right) \cdot p(\omega^{dis}) \cdot \eta_2 \tag{4.14}$$

For $\omega^{dis} < \omega_{th}$ we have the same case as for the linear velocity that is not affected by the driving disability. Hence $p\left(\omega^{ref}|\omega^{dis} < \omega_{th}\right) \sim \delta(\omega^{ref} - \omega^{dis})$, where $\delta(\cdot)$ is Dirac's delta function. The properties of the delta function imply that $p\left(\omega^{dis}|\omega^{ref}, \omega^{dis} < \omega_{th}\right) \sim \delta(\omega^{ref} - \omega^{dis})$ regardless of the prior distribution, $p(\omega^{dis})$.

If $\omega^{dis} = \omega_{th}$, any reference angular velocity ω^{ref} greater or equal than ω_{th} is feasible and therefore $p\left(\omega^{ref}|\omega^{dis} = \omega_{th}\right)$ is distributed uniformly in the interval $[\omega_{th}, \omega_{max}]$. Further, the prior $p(\omega^{dis})$ at the threshold velocity, ω_{th}, corresponds to a point mass represented by a Dirac delta function, i.e. $p(\omega^{dis} = \omega_{th}) \sim \delta\left(\omega^{dis} - \omega_{th}\right)$

By design, values of ω^{dis} larger than ω_{th} are impossible, i.e. $p(\omega^{dis} > \omega_{th}|\omega^{ref}) = 0$.

Taking these relations into account and normalizing the distribution leads to the performance model summarized in table 4.1.

TABLE 4.1: Angular velocity component of the performance model for a user with a weak left steering signal.

$p_{perf}\left(\omega^{dis}\|\omega^{ref}\right)$	$\omega^{dis} < \omega_{th}$	$\omega^{dis} = \omega_{th}$	$\omega^{dis} > \omega_{th}$
$\omega^{ref} \epsilon [\omega_{th}, \omega_{max}]$	0	$\delta(\omega^{dis} - \omega_{th})$	0
$\omega^{ref} \epsilon [\omega_{min}, \omega_{th}]$	$\delta(\omega^{dis} - \omega^{ref})$	0	0

Applying the obtained performance model to equation 4.11 leads to the following expression of the user model in the angular velocity for a user with a weak left signal (see also figure 4.7):

$$p_{user}\left(\omega_k^{dis}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) =$$

$$K_\omega \cdot \delta(\omega_k^{dis} - \omega_{th}) \cdot \left[\mathbb{G}\left(\frac{\omega_{max} - \omega_u^M}{\sigma_\omega}\right) - \mathbb{G}\left(\frac{\omega_{th} - \omega_u^M}{\sigma_\omega}\right)\right]$$

$$+\Pi(\omega_k^{dis} \epsilon [\omega_{min}, \omega_{th}]) \cdot \frac{K_\omega}{\sqrt{2\pi}\sigma_\omega} \cdot \exp\left(-\frac{\left(\omega_k^{dis} - \omega_u^M\right)^2}{2\sigma_\omega^2}\right) \tag{4.15}$$

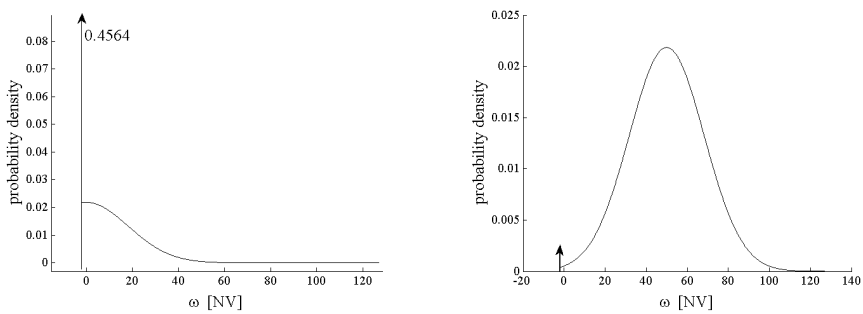
where:

1. ω_k^{dis} is the current noisy input of the user including the steering disability.
2. ω_k^M the angular velocity predicted by the ideal user model of equation 4.7 for a given intention.
3. K_ω is a normalizing constant.
4. $\mathbb{G}(\cdot)$ is the cumulative distribution function of standard normal random variable $\mathcal{N}(0, 1)$.

5. $\Pi(\omega_k^{dis}\epsilon[\omega_{min}, \omega_{th}])$ is the Heaviside step function in the interval $\omega_k^{dis}\epsilon[\omega_{min}, \omega_{th}]$.

Please note that for $\omega^{ref}\epsilon[\omega_{th}, \omega_{max}]$ all the user model's probability weight is condensed in one single value, namely ω_{th}. Considerable information has been eliminated from the user's signal through this driving disability. This user can no longer turn left, if ω_{th} is sufficiently small.

Figure 4.7 shows the user model for plans requiring an angular steering signal centre-left and to the right. The user model distribution tends to be more peaked at the threshold angular velocity, ω_{th}, the further the predicted reference intentional velocity, ω_k^M, lies to the left.



(a) Steering forward or slightly to the left.

(b) Steering to the right.

FIGURE 4.7: User model for a user with a weak left steering signal.

For the linear velocity, the user model corresponds to the first-principles reference user model discussed in section 4.3.1.

$$p_{user}\left(v_k^{dis}|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) = \frac{1}{\sqrt{2\pi}\sigma_v} \cdot \exp\left(-\frac{\left(v_u^{dis} - v_k^M\right)^2}{2\sigma_v^2}\right) \tag{4.16}$$

where v_k^M is the linear user velocity predicted by equation 4.7 for user plan $\boldsymbol{i}_k^M$.

We present the plan recognition results for this user with a weak left steering signal together with the shared control results in section 5.3.

Drawbacks of the performance model. Despite its theoretical interest to compare performance, the *performance model* is difficult to obtain in practice from driving data in reality. (Vanacker, 2008) proposes the *portal association technique*, which can generate simulation data to calibrate a performance model. In simulation, a portal moves along with the robot. The user then tries to maintain the robot in the middle of the portal, which changes its movement at regular intervals.

Another difficulty of the approach resides in the marginalisation of equation 4.11. Computing the integral analytically is only possible in rare cases. The moment the performance model considers non-stationary effects or is modelled as a non-Gaussian distribution, it will probably be necessary to approximate the integral. This is an additional source of modelling uncertainty, which reduces the quality of the user model.

In the next section we follow a different approach. Instead of the intermediate step of modelling the user's performance, we propose a general technique, which directly extracts arbitrary user models from real driving data. Further, it can deal with high-dimensional user plan representations without imposing the linearity limitation of the first-principles user model of section 4.3.1.

4.3.3 Gaussian Process user model with local trajectories

User modelling requirements

In contrast to the analytical derivation of the performance and user models explained previously, this section proposes the use of a non-parametric learning technique for user modelling. The goal is to extract the user model out of driving data directly without the intermediate step of a reference model.

The learning technique should adhere to the following list of requirements in order to be compatible with the proposed probabilistic framework for plan recognition:

Regression the task of creating a user model is a regression task because a continuous variable, the user input $\boldsymbol{u} = (v, \omega)$, has to be estimated.

Probabilistic the user model is a probability distribution in user-input space. It relates the likelihood of the present user inputs to a certain user plan and a previous history of robot actions and user inputs. Hence, the selected learning technique should directly output a probability distribution in function of the user plan and the previous history. This probability distribution must be compatible with the probabilistic framework for plan recognition described in section 3.2.

Informative the learning technique should yield a user model which allows distinguishing plan hypotheses based on the input of the user, whenever this input contains sufficient information. Conversely, completely random inputs should yield a model which does not introduce spurious information.

Analytical the selected learning technique should allow to understand how humans drive an electrical wheelchair. Therefore, the chosen learning technique has to provide information about which parameters have a higher influence when controlling a wheelchair.

Automatic parameters of the learning technique should be adapted automatically.

General the learning technique should have good generalization properties to unseen examples and provide a mean to asses the confidence in the predictions it is making.

Online in case that the user's driving performance varies, i.e. the confidence in the prediction drops consistently, it should be possible to adapt the parameters of the algorithm given the newly observed driving data.

Efficient a frequent drawback of non-parametric learning algorithms is their low efficiency, both in time and in storage space. In our framework for plan recognition and shared control, the distribution over user plans, $p_k\left(\boldsymbol{i}_k | \boldsymbol{u}_k, \boldsymbol{\mathcal{H}}_{1:k}\right)$, is updated at the information processing frequency of the

user. As such, frequent evaluations of the trained model are necessary, which have to be very efficient. Additionally, the driving data of the user should be compressed to a meaningful representation in order to avoid storing all previously seen examples.

Gaussian Process User Model

The user model, p_{user}, captures the likelihood of present user signals given the user has plan $\boldsymbol{i}_k$ in mind, and given a prior history of user signals, robot actions and sensor information. The product rule allows to factor the user model into a linear and angular component, as stated in equation 4.17.

$$p_{user}\left(\boldsymbol{u}_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) = p_{user}\left(v_k^u|\omega_k^u, \boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) \cdot p_{user}\left(\omega_k^u|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) \tag{4.17}$$

User modelling hypotheses We postulate a number of hypotheses in order to derive a computational model of wheelchair driving which fulfils the requirements stated previously. They constitute the prior knowledge the domain expert introduces to acquire a user model from real driving data. In the next table we make these assumptions explicit since they have an impact on the shape of the model. Then, we analyse each of the hypotheses in turn.

Path tracking. Humans reason about navigation tasks by tracking mental paths. We assume further, that their input to the robot provides some information about their plans, i.e. their signals, even if noisy, are not entirely random.

Gaussian likelihood. User inputs have a *Gaussian* distribution.

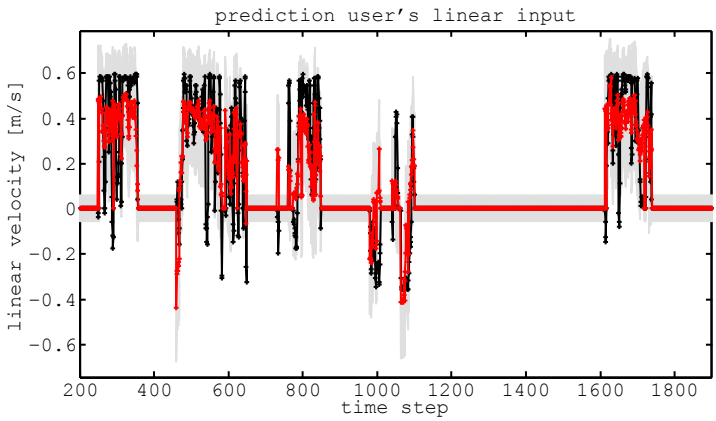
Parameterisation. Once the shape of the likelihood is selected, there is freedom in choosing the representation of user plans $\boldsymbol{i}_k$.

Mental via-points. The authors in (Demeester et al., 2006; Huntemann et al., 2007a; Demeester et al., 2008) assume that humans steer a wheelchair by employing perceptual cues placed on a mental path the user is tracking. Such perceptual cues are illustrated for instance in figure 4.5.

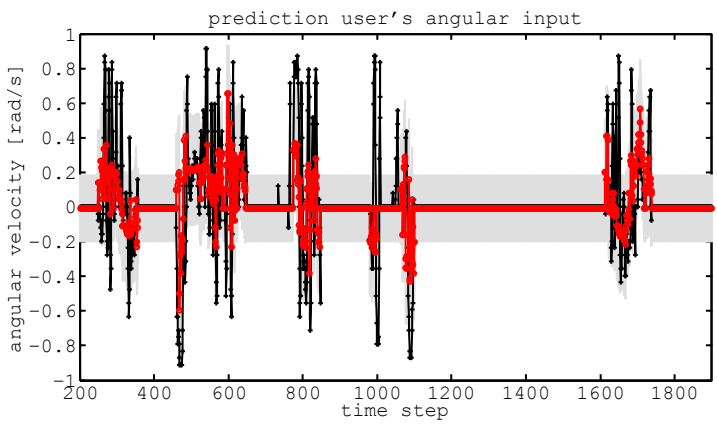
Independence. Throughout previous work (cf. (Demeester et al., 2008; Huntemann et al., 2008a)) we have assumed that the linear and angular user inputs are conditionally independent, given a certain plan hypothesis $\boldsymbol{i}_k$, i.e. $v_k^u \perp\!\!\!\perp \omega_k^u \mid \boldsymbol{i}_k$, with $\perp\!\!\!\perp$ denoting probabilistic independence and $\mid$ conditioning.

Equivalence. At equal driving conditions, i.e. equal amount of free space, the user drives backwards in the same way as forwards.

The path-tracking hypothesis is only briefly discussed here by anticipating some experimental results. Evidence supporting this claim consists of predicting the user's input based on a set of collision-free trajectories. Figure 4.8 shows the input of the user in black together with a mean prediction in red and one-standard deviation confidence interval in grey.



(a) linear velocity



(b) angular velocity

FIGURE 4.8: Prediction of the user's input by a set of collision-free trajectories.

We predict the input of the user in a post-experiment analysis with a Gaussian Process user model. This model receives at each time step the user plan, which most closely resembles the future trajectory of the user. It predicts then a mean velocity together with its standard deviation. Please refer to section 3.2.4 for a discussion about trajectory metrics to measure the proximity to the user's future path.

The evidence for the *path-tracking* hypothesis consists in the specific choice of plan parameterisation, which achieves reasonable predictions. The user's input is mostly within one standard deviation of the mean prediction, when we represent user plans as a set of tracking errors along a discretised ground-truth path.

The Gaussian likelihood hypothesis is chosen out of computational convenience. Nevertheless, it is supported by the empirical observation of the preceding section. Approximately tracking user plan $\boldsymbol{i}_k$ requires controls approximately matching the controls that could generate trajectory $\boldsymbol{i}_k$. A Gaussian likelihood is the most natural choice under these circumstances.

Parameterisation. As explained in section 2.2, a Gaussian Process, $\mathcal{GP}$, defines a distribution over functions in function space. In a Gaussian Process, function values form an infinitely uncountable collection of random variables with joint Gaussian distribution of any finite subset of them. Additionally, these function values map an arbitrary input space, $\mathcal{X}$ to the real line, $\mathbb{R}$. Therefore, Gaussian Processes, provide a rich Bayesian framework for regression, where learning is specified in terms of Gaussian probability density functions (*regression* and *probabilistic* requirements). Gaussian Process regression is non-parametric, i.e. the data is not summarised as an analytic model, but rather represents the model itself. Such representation covers a much richer class of functions than any parametric model. The designer still has some open choices when building the $\mathcal{GP}$ model, as will become clear in the ensuing discussion.

In our case, user plans and, eventually, a prior history, are the inputs to a Gaussian Process that determines the distribution over user inputs.

$$v_k^u \sim \mathcal{GP}_v\left(\mathbf{0}, cov_v(\boldsymbol{i}_k, \omega_k^u, \boldsymbol{\mathcal{H}}_{T-M:T})\right) \tag{4.18a}$$

$$\omega_k^u \sim \mathcal{GP}_\omega\left(\mathbf{0}, cov_\omega(\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T})\right) \tag{4.18b}$$

where $\sim$ denotes that the user input is distributed according to a specific $\mathcal{GP}$ with covariance function $cov_v(\cdot)$ or $cov_\omega(\cdot)$. Usually, the mean function of a Gaussian Process is considered zero, provided that the data is normalised by subtracting its mean.

The model builder still has the freedom to select the most appropriate and informative user plan representation. This corresponds to the *parameterisation* hypothesis. Choosing among equivalent representations should obey computational efficiency considerations (*efficiency* requirement). Of course, it is necessary to asses if representations are equivalent, at least to a certain extent.

The types of user plan hypotheses, which we considered in the experimental section, comprise circumferences of arcs of different curvatures. A specific example of the template of plans is displayed in figure 4.20. In principle, *any* dense set of collision-free hypotheses could serve to predict the user's plan in our framework.

For instance, (Demeester et al., 2008; Demeester et al., 2006), employ a geometric path planner to generate user plans.

We introduce two user plan representations. The first representation, which we call **path-tracking or $(\Delta L, \Delta\theta)$-parameterisation**, is generic and suited for any type of plan. ΔL is a signed displacement between the wheelchair pose, $\boldsymbol{x}_c = (x_c, y_c, \theta_c)$, and a discrete set of poses on the trajectory. The sign of each ΔL_i depends on the direction of motion, which is estimated by calculating the relative displacement along consecutive poses on the user plan. $\Delta\theta$ represents angular displacements normalised to the interval $]-\pi, \pi]$. Equation 4.19 summarises the most relevant components of the $(\Delta L, \Delta\theta)$-parameterisation.

$$\boldsymbol{i}_k = [(\Delta L_1, \Delta\theta_1) \cdots (\Delta L_M, \Delta\theta_M)]^T \tag{4.19a}$$

$$\Delta L_i = \text{sign}(x_{rel}) \cdot \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \tag{4.19b}$$

$$x_{rel} = (x_i - x_{i-1}) \cdot \cos(\theta_{i-1}) + (y_i - y_{i-1}) \cdot \sin(\theta_{i-1}) \tag{4.19c}$$

$$\Delta\theta_i = \angle(\theta_c - \theta_i) \in]-\pi, \pi] \tag{4.19d}$$

The second representation, the $(v, \omega, \Delta t)$**-parameterisation**, is especially appealing to parametrise circular arcs, but can also be extended to more general trajectories. These trajectories would be approximated as an arbitrary sequence of elementary circular arcs. Applying controls, (v, ω) for Δt results in a net displacement $(\Delta x, \Delta y, \Delta\theta)$ specified in the motion model of equations 4.20.

$$\Delta x = \begin{cases} \frac{v}{\omega}\Big(\sin(\omega\Delta t + \theta_c) - \sin(\theta_c)\Big) & v \neq 0, \omega \neq 0 \\ v\Delta t\cos(\theta_c) & \omega = 0 \end{cases} \tag{4.20a}$$

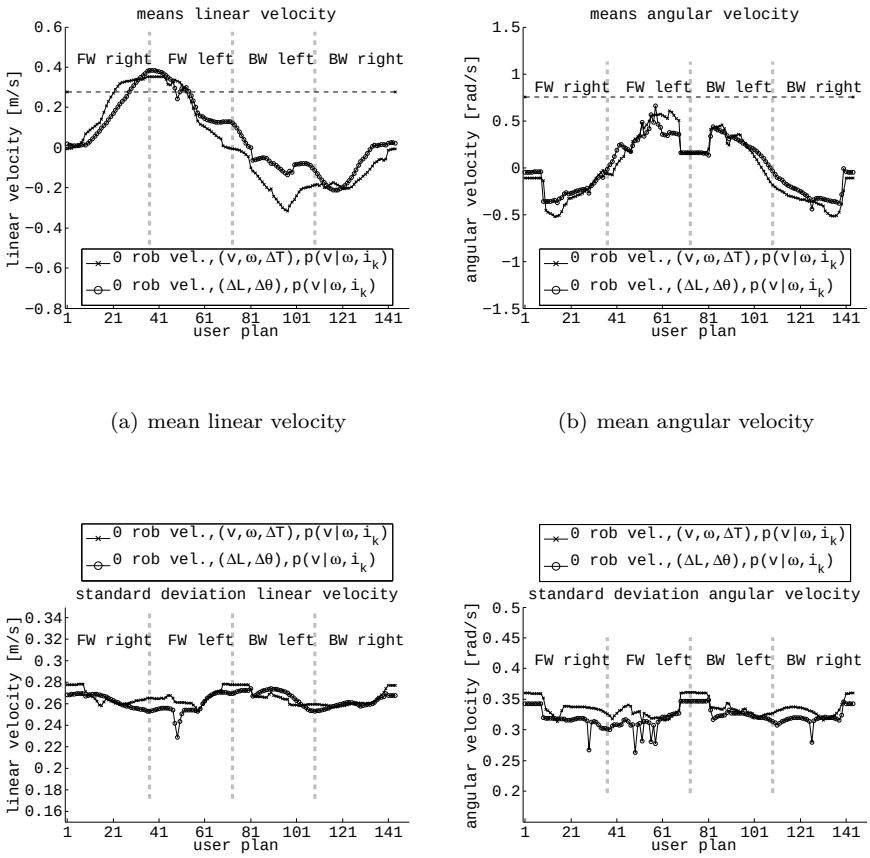
$$\Delta y = \begin{cases} \frac{v}{\omega}\Big(\cos(\theta_c) - \cos(\omega\Delta t + \theta_c)\Big) & v \neq 0, \omega \neq 0 \\ v\Delta t\sin(\theta_c) & \omega = 0 \end{cases} \tag{4.20b}$$

$$\Delta\theta = \begin{cases} \omega\Delta t & \omega \neq 0 \\ 0 & \omega = 0 \end{cases} \tag{4.20c}$$

The robot calculates the collision-free time, Δt_{free}, very efficiently[4] by placing obstacles in a precomputed grid. In our application, we derive the $(\Delta L, \Delta\theta)$-

[4] All collision-free trajectories of a user plan template with 144 plans are calculated in about $20 - 30ms$ on an Intel Core Duo Processor @$2.6GHz$ and $4GB$ of RAM.

parameterisation from the set of controls $(v, \omega, \Delta t_{free})$ by sampling the collision-free arcs uniformly.



(a) mean linear velocity

(b) mean angular velocity

(c) standard deviation linear velocity

(d) standard deviation angular velocity

FIGURE 4.9: Comparison of different user plan parameterisations without prior history of robot velocities. The figures show the estimated means and associated standard deviations for a set of 144 user plan hypotheses. The dotted horizontal line in the predictions of figure 4.9(a) and 4.9(b) represents the user's actual input $\boldsymbol{u}_k$. We can conclude that both parameterisations are very similar with respect to their predictions. Thus, the choice of a certain parameterisation should obey computational considerations.

Figure 4.9 compares the mean predictions and their associated standard deviations

for both parameterisations of $\boldsymbol{i}_k$ in a $\mathcal{GP}$-user model with 144 possible user plans. In the plot, plans are numbered according to the legend of figure 4.20. The predictions correspond to a specific snapshot of the experiments discussed in section 4.3.4. Figure 4.9 shows that both parameterisations lead to very similar results. An interesting effect in the angular velocity estimation of figure 4.9(b) is the small indentation on high curvature left-turns (plans around 69 to 80). This indentation introduces a bias in the angular component of the user model. We refer the reader to a similar situation represented in figure 4.53 of the experimental section for a more in-depth discussion of this bias.

The $(v, \omega, \Delta t)$-parameterisation is slightly more efficient because of a more succinct representation of user plans[5]. The appeal of the $(\Delta L, \Delta\theta)$-parameterisation resides in its analytical power. We will turn to this parameterisation again concerning the via-points hypothesis.

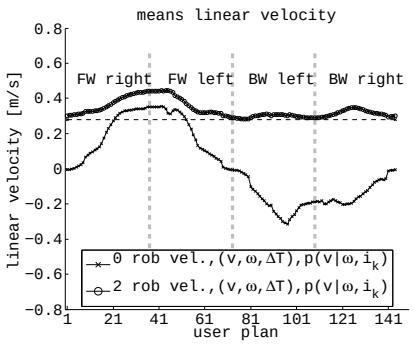
Extending the plan parameterisation with prior robot velocities. In figure 4.10 we compare the impact of extending the user plan representation with a history of prior robot velocities. Including prior velocities reduces the uncertainty of the user model. The standard deviation is lower for both the angular (figure 4.10(d)) as for the linear (figure 4.10(c)) predictions. However, the mean prediction is less informative. Figure 4.10(a) shows that the linear velocity prediction is almost constant across user plans.

In the following we will argue why the less informative user model with a robot action history of two velocities is wrong. Further, this analysis supports the hypothesis of section 3.2.2 that user signals $\boldsymbol{u}_k$ are conditionally independent of robot actions $\boldsymbol{a}_k$, given the user's current plan $\boldsymbol{i}_k$.

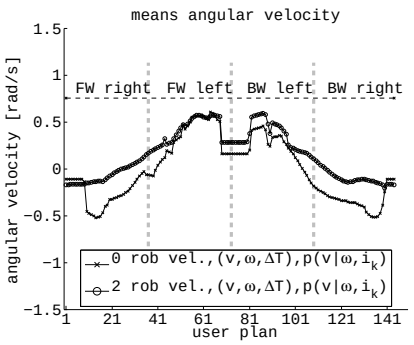
The user model with an robot action history of two velocities leads to almost equally probable forwards and backwards plans, as figure 4.11(a) indicates. Hence, this user model fails to distinguish between user plans. The reason for the model's poor performance is that the predicted linear velocity input, figure 4.10(a), is fairly constant regardless of the user's plan. This means that the user would point her joystick forwards, even if she intents to go backwards. The only determinant factor is the robot speed and not her intention. Clearly, the user model with an action history of two velocities fails the informative user modelling requirement.

The previous reasoning about robot velocities does not exclude plan-specific velocity profiles from the plan representation, as long as they differentiate intentions. Users might choose to track a geometric path at various speeds and then such extension of the plan representation is desirable.

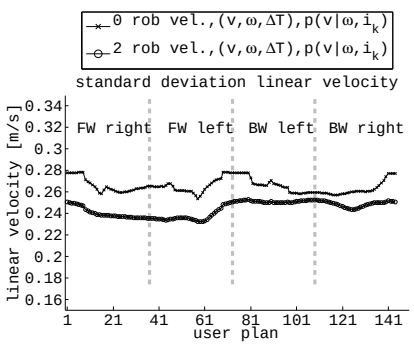
[5]On an Intel Core Duo Processor @2.6GHz and 4GB of RAM, the mean calculation time for a user model with 144 plans parametrised as $(v, \omega, \Delta t)$-triples is 360ms compared to 440ms for the $(\Delta L, \Delta\theta)$-parameterisation. In both cases, the $\mathcal{GP}$ user model contains 100 data points. These computation times could be further reduced by considering only dynamically feasible paths instead of the complete set of hypotheses.



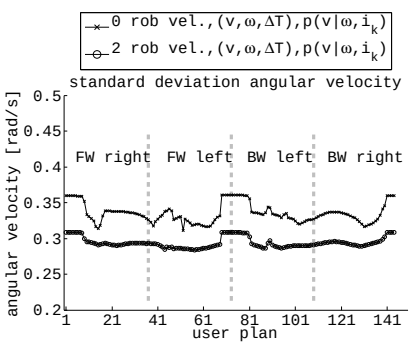
(a) mean linear velocity



(b) mean angular velocity



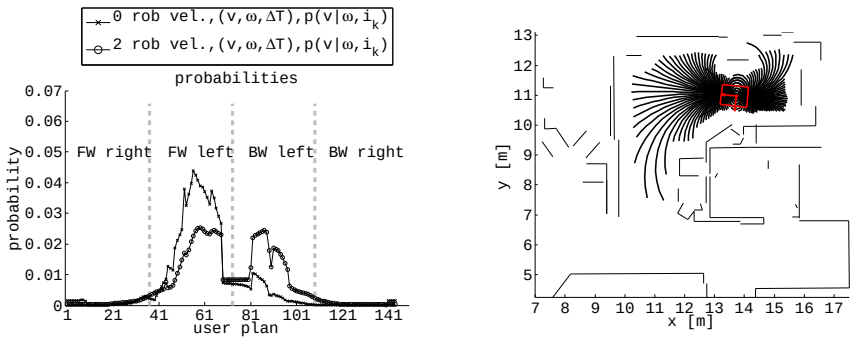
(c) standard deviation linear velocity



(d) standard deviation angular velocity

FIGURE 4.10: Impact of considering past robot velocities in the $\mathcal{GP}$-user model. The figures show the estimated means and associated standard deviations for a set of 144 user plan hypotheses. The dotted horizontal line in the predictions of figure 4.10(a) and 4.10(b) represents the user's current input $\boldsymbol{u}_k$. We observe that the model's uncertainty is reduced by including prior robot velocities, but its predictions are worse. The predicted linear velocity in figure 4.10(a) is independent of the user's plan, which contradicts informative user modelling requirement.

The via-points hypothesis (Demeester et al., 2006; Huntemann et al., 2007a; Demeester et al., 2008) assumes that humans steer a wheelchair by tracking visual cues on a mental path such as those in figure 4.5. The $\mathcal{GP}$-user model no



(a) user model

(b) environment

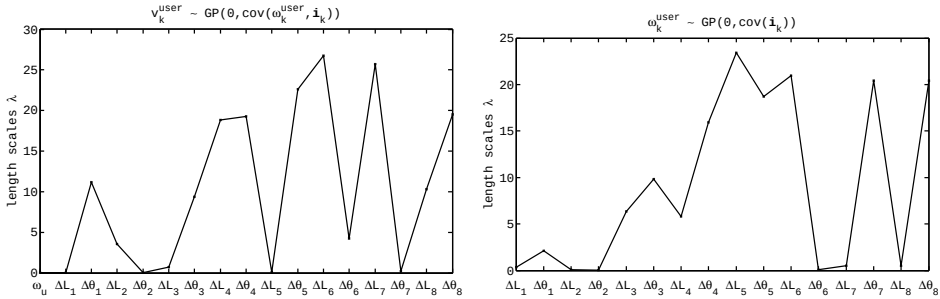
FIGURE 4.11: Figure 4.11(a) shows the probability distribution over a set of dense user plans (figure 4.11(b)) according to a $\mathcal{GP}$-user model that includes past robot velocities and a user model without prior robot history. Forwards and backwards leading plans are equally probable for the user model with a history of robot actions. This means that the user would point her joystick forwards even if she intends to go backwards. Clearly, such model is undesirable as it fails the informative user modelling requirement.

longer requires this assumption. However, the $\mathcal{GP}$ user modelling results provide supporting evidence for the via-points hypothesis.

Gaussian Process is a non-parametric regression technique, where target functions are represented implicitly by the data. Nevertheless, a number of hyper-parameters[6] control the shape of the regression function. The hyper-parameters of our $\mathcal{GP}$-user model are optimised from real data via Automatic Relevance Determination (ARD). ARD fulfils the requirement of *automatic* parameter adaptation. It produces the relative weights of the inputs (the length-scale matrix P) in the computation of the user model. Hence, it enables to analyse which input dimensions are significant for a particular user when steering a wheelchair (*analytic* requirement).

According to (Rasmussen and Williams, 2006), the length scales indicate roughly the distance you need to move along a certain direction in input space in order to achieve significant variation at the output. Large length scales compared to the range of the input direction mark thus irrelevant directions, whereas small length scales indicate input directions to which the output is especially sensitive.

[6] A hyper-parameter involves a second level of abstraction when building a model. First-order parameters define the model directly, whereas hyper-parameters parametrise the primary parameters.



(a) length scales linear velocity (b) length scales angular velocity

FIGURE 4.12: Length scales of $\mathcal{GP}$-user model where user plans are represented in $(\Delta L, \Delta\theta)$-parameterisation. Large length-scales correspond to irrelevant input dimensions, whereas low length scales indicate input dimensions to which the output function is especially sensitive.

Figure 4.12 displays the results of Automatic Relevance Determination for a $\mathcal{GP}$-user model. User plans, $\boldsymbol{i}_k$, are represented by the $(\Delta L, \Delta\theta)$-parameterisation introduced previously. These plans are sampled maintaining equal displacement along the trajectory. In order to keep the input dimension constant, shorter paths are extended by repeating the last pose.

We can identify two regions of significant input in the length-scale plots of figure 4.12. It is noteworthy that length scales of inputs $\Delta\theta_2$ and ΔL_3 for the linear velocity component (figure 4.12(a)), and of inputs ΔL_2 and $\Delta\theta_2$ for the angular velocity component (figure 4.12(b)), are very low compared to surrounding dimensions. Those other dimensions are redundant in the user model predictions. Somewhere between point two, $(\Delta L_2, \Delta\theta_2)$, and three, $(\Delta L_3, \Delta\theta_3)$, lies probably a cue, which the user employs to steer the wheelchair.

The second region of significant input resides in the last set of inputs, both for the angular as for the linear components. For the linear $\mathcal{GP}$-model, the significant inputs are ΔL_5 and $\Delta\theta_7$. The angular component is especially sensitive to dimensions $\Delta\theta_6$, ΔL_7, and ΔL_8. Here the notion of a cue which predicts both the angular and linear inputs of the user is more diffuse.

Figure 4.12 also reveals that the $(\Delta L, \Delta\theta)$-parameterisation contains considerable redundancy. This is a further argument in favour of the more concise $(v, \omega, \Delta t)$ user plan representation.

The independence hypothesis allows to simplify the user model factorisation in equation 4.17 as follows:

$$p_{user}\left(\boldsymbol{u}_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{1:k}\right) = p_{user}\left(v_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{1:k}\right) \cdot p_{user}\left(\omega_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{1:k}\right) \tag{4.21a}$$

$$v_k^u \sim \mathcal{GP}_v\left(\mathbf{0}, cov_v(\boldsymbol{i}_k)\right) \qquad \omega_k^u \sim \mathcal{GP}_\omega\left(\mathbf{0}, cov_\omega(\boldsymbol{i}_k)\right) \tag{4.21b}$$

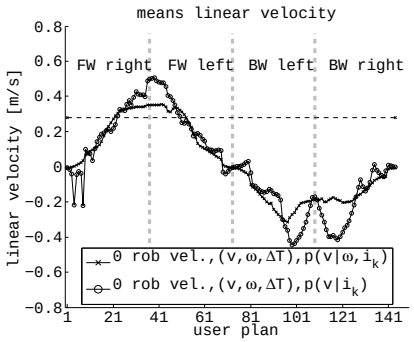
In the independent user model, the user's linear input, v_k^u, no longer depends on her angular input, ω_k^u. Figure 4.13 compares the predictions of dependent and independent $\mathcal{GP}$-user models, where dependency refers to the user's input. The mean angular predictions, figure 4.13(b), are very similar in both cases, since the angular component of both models is based on the same inputs. The covariance function underlying both angular components depends only on the plan representation, $\omega_k^u \sim \mathcal{GP}_\omega\left(\mathbf{0}, cov_\omega(\boldsymbol{i}_k)\right)$. Variations, such as those of the angular component's standard deviation (figure 4.13(d)) can be attributed to convergence of the training procedure to slightly different solutions.

The main difference between independent and dependent user models resides in the linear predictions, figure 4.13(a). These predictions are moderately more extreme in the independent case. The dependent linear component considers that the user's linear input remains moderate at intermediate angular velocities, since this particular user prefers to turn at low speeds.

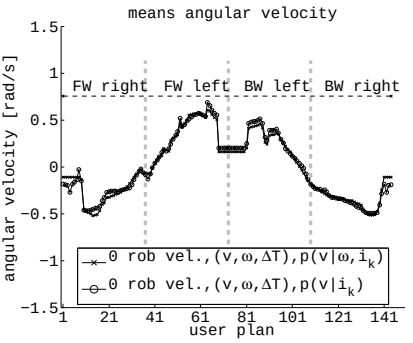
Figure 4.14 displays the dependent and independent user model distributions incorporating the mean predictions and the standard deviations of figure 4.13. Both user models are virtually indistinguishable for the present situation. The independence hypothesis is a reasonable assumption as long as the user plan representation, $\boldsymbol{i}_k$, is sufficiently rich to uncouple both components of the user input.

This hypothesis is further supported by the arrangement of linear and angular controls on a conventional joystick. The user can control both axes independently. They only become linked on the borders of the restrictor plate (cf. figure 5.1 in section 5.1.1 for examples of joystick restrictor plates), where the inputs are limited in hardware in order to avoid turning at high linear speed. Therefore, outside of the joystick's border, we can assume that both inputs are independent given the user's plan.

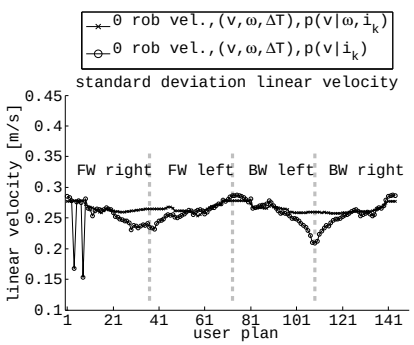
The equivalence hypothesis is introduced out of necessity, rather than derived from observed driving behaviour. During the experiments described in section 4.3.4 it was relatively difficult to gather sufficient data of backwards driving. The user's limited visibility towards the wheelchair's back made driving backwards too unsafe. However, user modelling and plan recognition still have to distinguish between forwards and backwards plans. Therefore, out of convenience, we assume that under the same driving conditions, i.e. with the same amount of free space, the user would drive equally backwards as she would drive forwards.



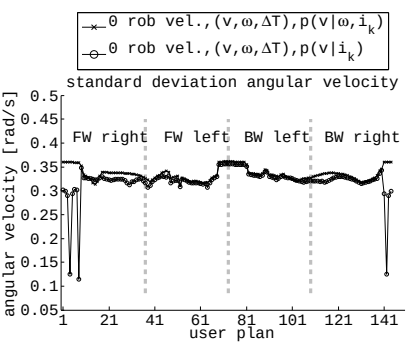
(a) mean linear velocity



(b) mean angular velocity



(c) standard deviation linear velocity



(d) standard deviation angular velocity

FIGURE 4.13: Comparison of two $\mathcal{GP}$-user models, one assuming dependent and the other assuming independent linear and angular user inputs. The figures show the estimated means and associated standard deviations for a set of 144 user plan hypotheses. The dotted horizontal line in the predictions of figure 4.13(a) and 4.13(b) represents the user's current input $\boldsymbol{u}_k$. The linear predictions in figure 4.13(a) are less extreme at low-to-intermediate curvatures for the user model with dependent inputs. This indicates that the user prefers to turn at low speeds.

Even if this hypothesis might be contentious, we believe it is the least damaging information we can introduce. It differentiates forwards from backwards plans when the user is driving forwards. In the experimental section we show that, even for

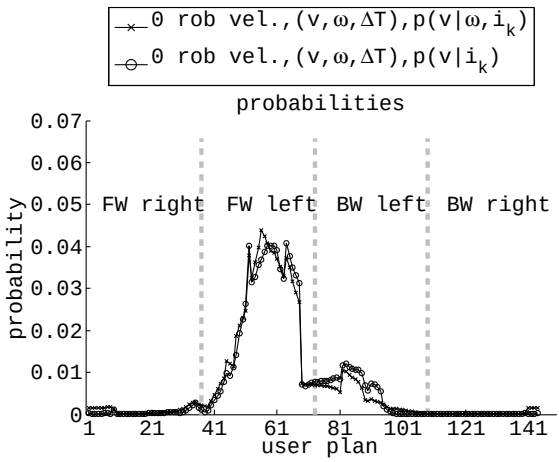


FIGURE 4.14: $\mathcal{GP}$-user model distributions when considering independent and dependent user inputs. Both user models are almost indistinguishable, which supports the independence hypothesis.

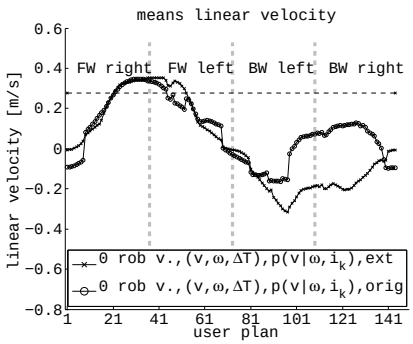
the few backwards-driving situations, the user model makes reasonable predictions.

The equivalence assumption extents the original dataset with virtual driving data. These data comprise a sign-inverted linear user input matching the same available free space in the opposite direction. In the $(v, \omega, \Delta t)$-parameterisation only the linear velocity's sign needs to be inverted. The tracking-error $(\Delta L, \Delta\theta)$-parameters require sign inversions of all displacements ΔL_i along the trajectory, as well as angular inversions of all $\Delta\theta_i$-parameters by adding π to each angular displacement.

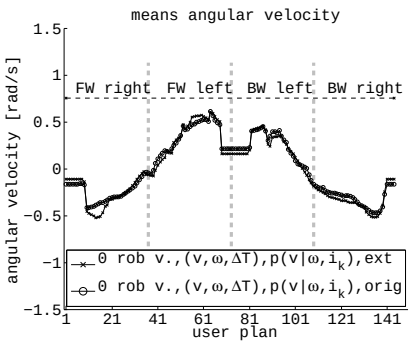
Figure 4.15 compares linear and angular velocity predictions for the original dataset as well as for a dataset extended by 50% of virtual backwards driving data. In both cases plans are represented in the $(v, \omega, \Delta t)$-parameterisation. The main difference appears in the linear velocity predictions, figure 4.15(a). In the extended dataset forwards and backwards plans are clearly differentiated, with backwards-moving plans resulting in negative velocity predictions.

The price for better performance of the extended dataset is a slight increase of the overall uncertainty, i.e. higher standard deviation in figure 4.15(c). Angular velocity predictions in figure 4.15(b) remain unaffected from the dataset extension. No additional information is introduced concerning the angular user model component.

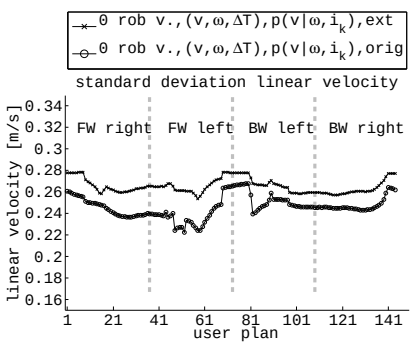
Figure 4.16 displays the user model distributions for the original and extended



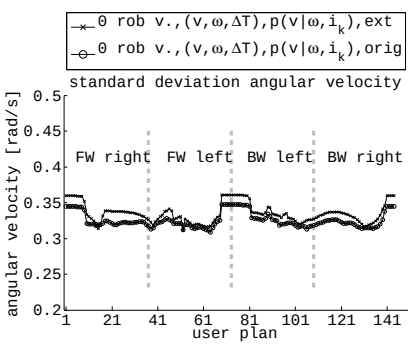
(a) mean linear velocity



(b) mean angular velocity



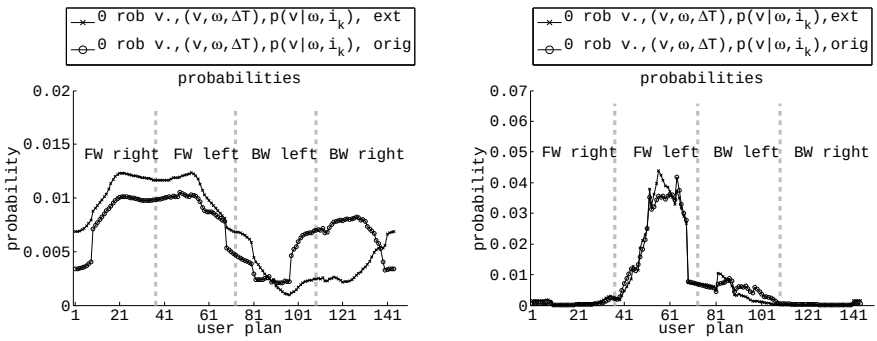
(c) standard deviation linear velocity



(d) standard deviation angular velocity

FIGURE 4.15: Impact of extending user modelling data set with negative velocities. The figures show the estimated means and associated standard deviations for a set of 144 user plan hypotheses. The dotted horizontal line in the predictions of figure 4.15(a) and 4.15(b) represents the user's current input $\boldsymbol{u}_k$. The main difference between both datasets resides in the linear velocity predictions, figure 4.15(a). For the user model trained with the extended data, forwards and backwards plans are clearly differentiated. The price for better performance of this model is a slight increase in the overall uncertainty, i.e. larger standard deviations.

datasets. Although no significant difference appears in figure 4.16(b), which shows the result of combining angular and linear velocity components, bad linear



(a) linear component of the user model

(b) complete user model

FIGURE 4.16: User model distribution for an extended and the original dataset. In the linear component of the user model, figure 4.16(a) the original dataset fails to differentiate forwards from backwards advancing plans. Conversely, the extended dataset leads to a user model that already introduces a preference for forwards, given that the user's current linear input is around 0.3 m/s. However, the complete user model in figure 4.16(b) enhances the information content of the linear component by combining it with the angular user model component.

predictions can reduce the amount of information of the user model. For instance, in figure 4.16(a), we display just the linear component of the user model. For the original dataset forwards and backwards plan are almost undistinguishable. On the contrary, the extended dataset leads to a user model which already introduces a justified preference for forwards advancing arcs.

Summary of the $\mathcal{GP}$-user model and evaluation against the requirements

Summarising the previous section, we restate the core properties of the $\mathcal{GP}$-user model and give a schematic overview of how this model addresses the requirements of section 4.3.3. The set of equations 4.22 formulates the complete user modelling problem:

$$p_{user}\left(\boldsymbol{u}_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) = p_{user}\left(v_k^u|\omega_k^u, \boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) \cdot p_{user}\left(\omega_k^u|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right) \tag{4.22a}$$

$$v_k^u \sim \mathcal{GP}_v\left(\mathbf{0}, cov_v(\boldsymbol{i}_k, \omega_k^u)\right) \qquad \omega_k^u \sim \mathcal{GP}_\omega\left(\mathbf{0}, cov_\omega(\boldsymbol{i}_k)\right) \tag{4.22b}$$

Definition. The user model $p_{user}\left(\boldsymbol{u}_k|\boldsymbol{i}_k, \boldsymbol{\mathcal{H}}_{T-M:T}\right)$ describes the likelihood of present user inputs, given the user has a certain plan in mind. This likelihood needs to be calculated for every single plan hypothesis $\boldsymbol{i}_k$ in the plan library. As explained in section 3.2, the instantaneous plan estimates of the user model are then combined with a prior over plans to reflect the temporal evolution of the user's intentions. In order to achieve user-specific plan estimates, the user model needs to be tailored to the user's driving style, preferably derived from real driving data.

If the user interface is continuous, which is approximately true for conventional wheelchair joysticks, it is possible to extract a continuous probabilistic mapping from user plans to user inputs. Conventional joysticks consist of two independently controllable axes, which define the linear and angular velocities respectively. Therefore, we can split the user model p_{user} into two components, one for the linear velocity input and another one for the angular input. These components can then be represented as two Gaussian Processes, $\mathcal{GP}_v$ and $\mathcal{GP}_\omega$, which are trained separately. A link between linear and angular components can be introduced by including either component into the other Gaussian Process.

Parameterisation of user plans. The next question concerns the parameterisation of user plans $\boldsymbol{i}_k$, which defines the Gaussian Process' input space. In the previous section we introduced two alternative representations for a path template consisting of arc circumferences, the tracking-error $(\Delta L, \Delta\theta)$-parameters and the $(v, \omega, \Delta t)$-parameters. As both parameterisations achieve equivalent performance, we opt for the more compact $(v, \omega, \Delta t)$-parameterisation. Additionally, we include a certain amount of past history into the plan representation, which consists of the plan's current extension, Δt_k, as well as the previous extension, Δt_{k-1}. Each user plan, $\boldsymbol{i}_k^M$, at time step k in the Gaussian process corresponds to $\boldsymbol{i}_k^M \equiv (v^M, \omega^M, \Delta t_k^M, \Delta t_{k-1}^M)$.

Calibration of the user model. The Gaussian Processes $\mathcal{GP}_v$ and $\mathcal{GP}_\omega$ are trained separately using an extended training set. Calibration or training of the $\mathcal{GP}$-user model means to adjust the model's hyper-parameters through Automatic Relevance Determination (ARD). It also requires to calculate the inverse covariance matrix representing the non-parametric Gaussian Process. Finally, different possible $\mathcal{GP}$ models can be compared in order to select the most appropriate. All three processes form the first model-building stage, which we performed with the Gaussian Process For Machine Learning (GPML) library of (Rasmussen et al., 2010).

Within the training data, the user inputs $\boldsymbol{u}_{1:T}$ form the target values that are mapped probabilistically to the ground-truth hypotheses $\boldsymbol{i}_{1:T}^{GT}$. Ground-truth represents the set of user plans most closely matching the future user path at each time step according to the metric described in section 3.2.4.

It is required to extend the data with a certain amount of virtual data because backwards-driving is too dangerous for the user and is therefore insufficiently represented in the original data. The user model has to perform well over the complete spectrum of plan hypothesis and distinguish forwards- from backwards-advancing plans.

In order to address the requirements of online adaptation (*online* requirement) and efficiency (*efficiency* requirement) we adopt an approximate Gaussian Process algorithm for user modelling. This algorithm performs a sparse Gaussian Process process approximation, which can be applied sequentially (online) to data (Csato and Opper, 2002). This second stage of model building was performed with the Online Gaussian Process (OGP) library of (Csató, 2008). Further, we evaluated the resulting $\mathcal{GP}$-user model in operation in *C++*.

Model comparison of different $\mathcal{GP}$-user models should obey quantitative and qualitative criteria. It is necessary to determine the optimal number of points included in the model, the optimal covariance function, or the most convenient user plan representation.

We compare different models in terms of their marginal likelihood[7] in the framework of Bayesian model selection (Rasmussen and Williams, 2006).

Bayesian model section is not the sole criterion to define the optimal user model. Qualitative criteria such as the model's informative value or its efficiency also play a significant role. For instance, the marginal likelihood of a user model that includes two prior robot velocities in the plan representation is larger than a user model without robot velocities in the plan representation. However, the performance of the latter model is better (cf. the analysis of the parameterisation hypothesis).

[7]The marginal likelihood is the probability of the data under a given model when the model's parameters are integrated out, i.e. $\log p\left(\boldsymbol{y}_{target}|\boldsymbol{X}_{data}\right)$ with $y = f(\boldsymbol{x}) + \epsilon$. For a Gaussian Process the model parameters correspond to the latent function values f, which are unobserved due to the process noise ϵ.

The marginal likelihood is useful when comparing across model classes such as different covariance functions for the same plan parameterisation.

Evaluation of the user modelling requirements. We conclude this section with an evaluation of the $\mathcal{GP}$-user model against the requirements of section 4.3.3 in table 4.2.

TABLE 4.2: Check-list of the requirements for user modelling

Requirement	**Explanation**	**Evaluation**
Regression	$\mathcal{GP}$ estimate continuous variables	✓
Probabilistic	$\mathcal{GP}$ is Bayesian framework for estimating functions probabilistically	✓
Informative	choice of parameterisation of user plans $\boldsymbol{i}_k$ increases informative power of the user model	✓
Analytical	$\mathcal{GP}$-user model allows analysing driving style. Please refer to the experimental section.	✓
Automatic	automatic relevance determination to select hyper-parameters	✓
General	this requirement is analysed in experimental section. In general, large standard deviations indicate that the model is performing worse.	✓
Online	online $\mathcal{GP}$ approximation allows incremental variation of user model	✓
Efficient	sparse $\mathcal{GP}$ approximation with succinct user plan representation. There is a trade-off between efficiency and representative power, namely how many data points conform the model.	$\pm$

4.3.4 Experiments: user modelling and plan recognition for a user diagnosed with spastic quadriplegia

Description of the user. Figure 4.17 shows the user participating in experiments within the EU-funded MOVEMENT project (Mayer et al., 2007). The user, who was 19 at the time of the experiments, is a female presenting symptoms of spastic quadriplegia. Quadriplegia refers to the fact that her condition affects all limbs.



FIGURE 4.17: User participating in the experiment at iRv/vilans trying to get out of the bathroom in a home-like environment.

Spasticity generally results from damage to the part of the brain that controls voluntary movement. It may also occur when the nerves travelling from the brain down to the spinal cord have been damaged (Kantor, 2007). In the case of the user participating in the tests, spasticity was caused by an accident when she was seven years old. As a consequence of her condition, she is not capable of steering

a conventional electrical wheelchair safely. During the experiments, collisions, especially in narrow locations, abounded. She clearly fits the profile of users who currently are not prescribed power wheelchairs and could benefit enormously from navigation assistance.



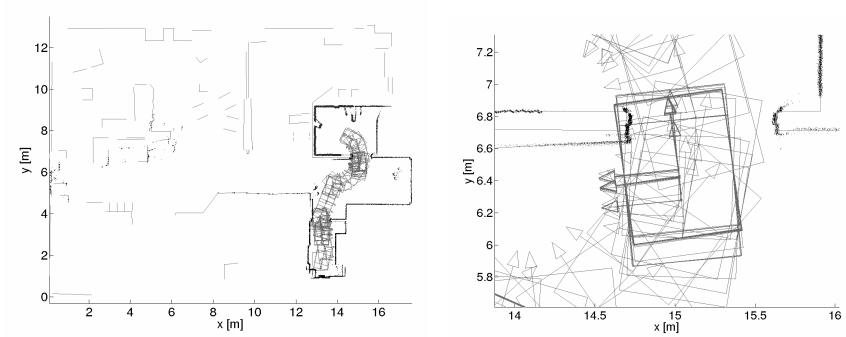
FIGURE 4.18: Typical hand-posture to hold the joystick as a consequence of spastic condition. This interface is not customised.

An additional factor complicating navigation was that the interface was not particularly adapted to the user, but rather a conventional joystick. Figure 4.18 shows how the user postured her hand in relation to the joystick. Another symptom of spasticity were involuntary movements, which caused frequent changes in overall posture, and thus, at times, reduced her field-of-view considerably. Despite the difficulties with fine-control of the wheelchair, this user was perfectly capable to reason about navigation tasks. However, she had to maintain a very high degree of concentration. Presumably, driving while being observed had also an impact on the user's stress level and, consequently, on her level of fatigue.

Experiment description. The experiments have been conducted in a home-like environment at iRv/vilans in Hoensbroek, The Netherlands (cf. figure 4.21 for a map of the environment). We collected two datasets on two different days. One dataset will be employed for training a $\mathcal{GP}$-user model and the other to evaluate its performance.

The training dataset lasts about 9 minutes during which the user had to solve different navigation tasks. No driving assistance was provided since the purpose of the experiment is to study the user's driving style. Typical manoeuvres include most of the situations encountered in daily-life such as such as driving forwards, crossing narrow openings, turning at different radii, and also executing more complex trajectories to reach far-away poses in the environment. One example

manoeuvre is shown in figure 4.19(a). As figure 4.19(b) reveals, collisions with the environment happened occasionally.



(a) Example manoeuvre (b) Points of collision

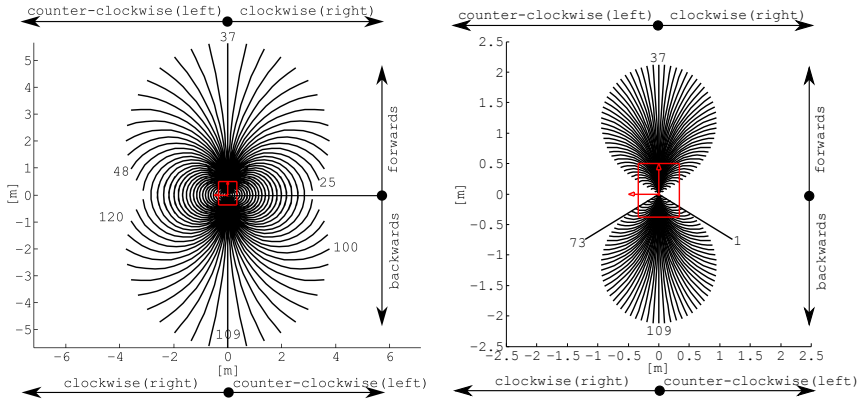
FIGURE 4.19: Segment of trajectory contained in the training set. Given the large dimensions of the wheelchair compared to the environment, it is very difficult for the user to control it without colliding.

The data reported in the remainder of this section concerns exclusively the test dataset, which did not contribute to the calibration of the user model.

Description of user plans. Figure 4.20 shows different extensions of the path template, which represent a reasonably dense set of potential user plan hypotheses. These paths are centred on the robot and adapted to the local environment for each evaluation of the user model.

The numbers in figure 4.20 appear on all graphs that plot probability distributions as a function of plan numbers.

- plan 1: turn on-the-spot clockwise (right turn).
- plans $2 - 36$: advancing forwards on a circular arc and turning clockwise (right). Increasing plan number denotes a higher forward speed and a lower magnitude angular speed. For these plans the angular speed is negative. The curvature decreases with increasing plan number.
- plan 37: advancing straight-ahead.
- plans $38 - 72$: advancing forwards on circular arc while turning counter-clockwise (left). Increasing plan number means a lower forward speed and higher angular speed. For these plans the angular speed is positive. The curvature increases along plan numbers.



(a) long paths

(b) short paths

FIGURE 4.20: Local paths with legend. The straight lines (plans 1 and 73) in figure 4.20(b) represent the final orientation of the robot after turning on-the-spot either clockwise or counter-clockwise.

- plan 73: turn on-the-spot counter-clockwise (left turn).
- plans $74 - 108$: moving backwards while turning counter-clockwise (left). Increasing plan number means higher magnitude backwards speed and lower magnitude angular speed. For these plans the angular speed is positive and the linear speed negative. Curvature decreases for higher plan numbers.
- plan 109: moving backwards on straight line
- plans $110 - 144$: moving backwards while turning clockwise (right). Increasing plan number means lower magnitude backwards speed and higher magnitude angular speed. For these plans both the angular and the linear speeds are negative. Curvature increases with plan number.

The following sections describe the most salient situations in the test dataset according to the models for plan recognition and user modelling calibrated from the training dataset.

From corridor to the kitchen

In this section we analyse the trajectory of the user when driving from a starting pose in the corridor to the kitchen in iRv/vilans' home-like environment. The complete trajectory is shown in figure 4.21 and the user's input in figure 4.22.

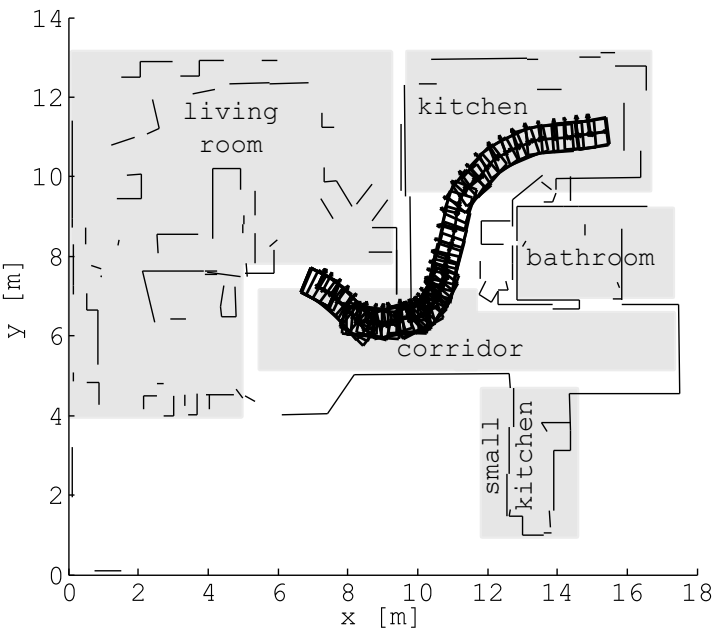


FIGURE 4.21: User's trajectory for reaching the kitchen from a starting pose in the corridor.

The user model distribution corresponding to the complete manoeuvre is displayed in figure 4.23 and the posterior over user plans, as computed with the formalism of section 3.2, in figure 4.24.

Our performance benchmark for plan recognition and user modelling is the concentration of probability mass around ground-truth. Visual inspection of the user model (figure 4.23) shows a reasonable convergence to ground-truth, which is even higher for the posterior probability (figure 4.24).

We have highlighted certain situations in the probability plot marked by dashed vertical lines. These situations will be described in detail because they are either representative of the driving style of the user or, in some cases, represent challenging

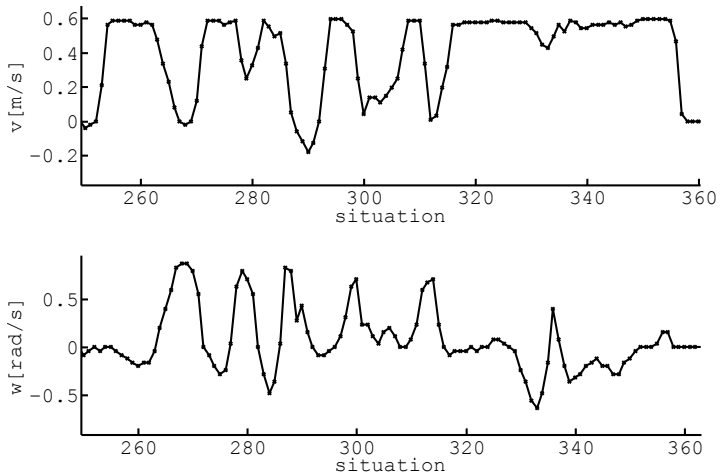


FIGURE 4.22: User's input while driving from a starting pose in the corridor to the kitchen. The sampling time is $\Delta T = 0.3\ s$.

modelling situations. In the detailed description we always provide a set of four plots. The first plot shows the dense path template (figure 4.20) for the current situation. Next, we display a probability plot where the prior, user model and posterior probabilities are represented in function of plan number according to the legend of figure 4.20. The last two graphs translate the probability graph into a spatial representation of the user model and the posterior. More probable trajectories appear darker in these spatial plots. The future path of the user, which is coloured in blue[8] in the spatial distribution plots, serves as reference for user model and posterior predictions.

In figure 4.25 the user is advancing on a right-turning arc of low curvature during situation 262. According to the user model (figure 4.25(c)), the highest likelihood trajectories involve right turns of low curvature matching the present user input. However, as the angular velocity input of the user is only slightly negative, the forward straight-line and left arcs of intermediate curvature maintain probability mass. The lowest likelihood is assigned to right turns of high curvature. All remaining plans have negligible likelihood. The posterior in figure 4.25(d) is considerably more focused around the forward straight-line and low curvature arcs to the left and right.

Situation 265 (figure 4.26) shows the user starting to turn left. While increasing her angular velocity, the user is reducing her linear velocity input. Therefore, the

[8]The future path of the robot is coloured in blue in the electronic version of the text. In the printed edition, the future path appears in darker colour and is normally clearly distinct from the user plan library. It is still possible to evaluate if the spatial plots converged correctly if all dark trajectories coincide. However, in figures 4.65(c) and 4.70(c) we showed the future path in light blue to improve visibility.

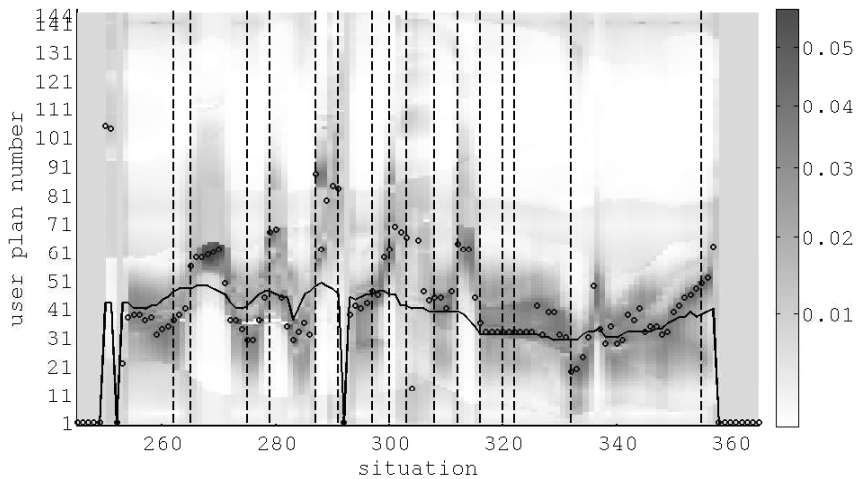


FIGURE 4.23: User model for driving from a starting pose in the corridor to the kitchen. Darker grey values are proportional to higher user plan likelihoods. The circles indicate the maxima of the distributions at each time step. Further, a continuous line marks ground-truth, which corresponds to the user plan which is closest to the future trajectory at each situation. Proximity is measured via the Hausdorff's metric (cf. section 3.2.4). Dashed vertical lines mark the situations described in the text.

most likely trajectories in the spatial user model plot, figure 4.26(c), correspond to left turning arcs of intermediate curvature. The user model likelihood is centred on intermediate curvatures and gradually decreases along left turns of low curvature on one side and high curvature on the other side, cf. figure 4.26(b). Due to the low linear input, turning backwards to the left maintains around half the maximal likelihood. Some residual likelihood remains in high curvature turns to the right, both for forwards as for backwards plans. This stems from the fact that the user's angular velocity is not yet maximal, and negative angular velocities are still within 1.5 standard deviations of the user's input. The posterior distribution reduces the uncertainty of the user model considerably and forward trajectories turning left are among the most probable (figure 4.26(d)). The preference for forward trajectories in the posterior shows the effect of spatio-temporal filtering achieved through plan recognition. The user's plan is derived from her time-varying input as well as through spatial reasoning about the environment. Past user behaviour, namely driving forward, is combined with the user model to obtain the current posterior

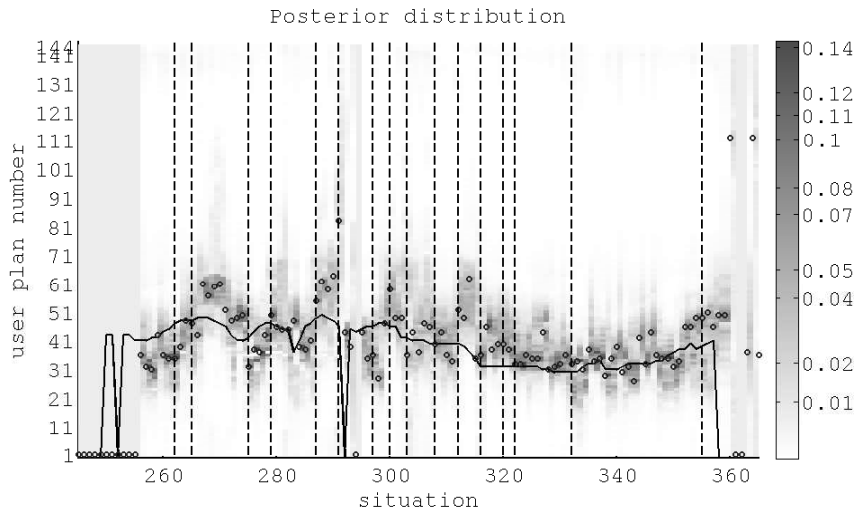
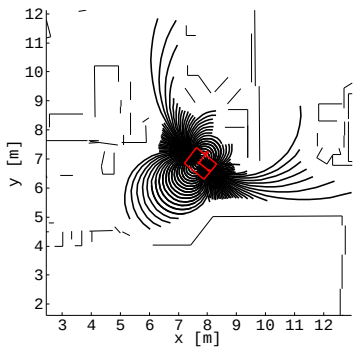


FIGURE 4.24: Posterior distribution over user plans for the manoeuvre in figure 4.21.

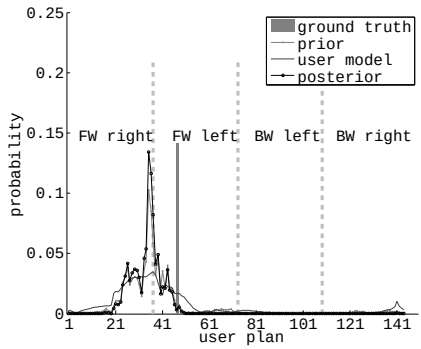
over user plans. In figure 4.26(b), the prior over user plans, which summarises past driving, has its peak on forward right-turning arcs of low curvature and the straight-line hypothesis. Forward left-turns of low curvature are less probable a priori as the wheelchair is still turning right during situation 265. The resulting posterior, which combines prior and user model, reflects the user's preference for turning left, but still assigns some confidence to right turns that represent past driving behaviour.

Figure 4.27 displays a situation with a multi-modal user model. Although the long-term plan of the user is among the most likely hypotheses, right turning arcs have the highest likelihoods. The user intends a left turn on the long term, but she fails to maintain a straight course and overshoots slightly by pointing her joystick right. The multi-modality of the user model originates from a discontinuity in the estimation of the mean angular velocity for certain plans. Comparing probable trajectories of user model (figure 4.27(c)) and posterior (figure 4.27(d)), one observes similar distributions with a more peaked posterior. The probability concentrates in the posterior around one mode of the user model, but plans in the secondary lobe still maintain probability mass, as indicated in figure 4.27(b).

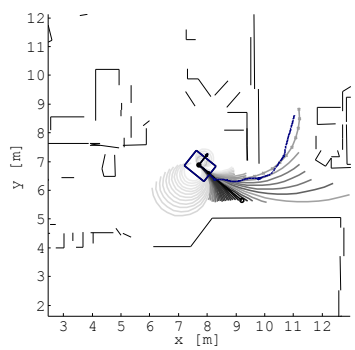
In situation 279, figure 4.28, the user model assigns considerable probability mass to plans off ground-truth. The most likely trajectories are left-turns of high curvature



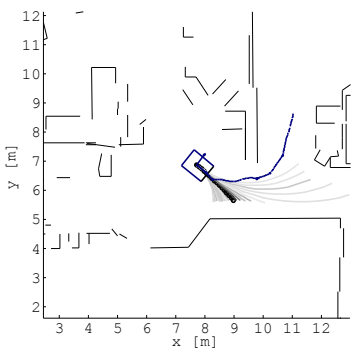
(a) all trajectories



(b) probabilities



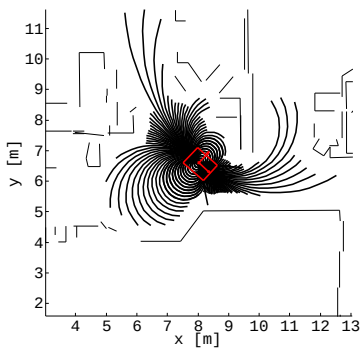
(c) user model



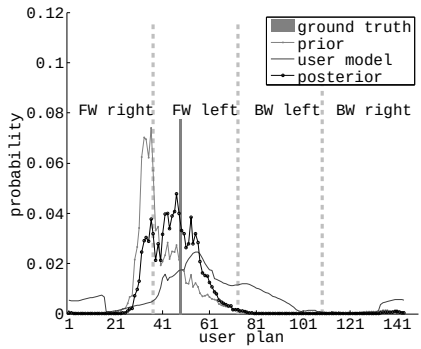
(d) posterior

FIGURE 4.25: Advancing on a right turn of low curvature at time step 262.

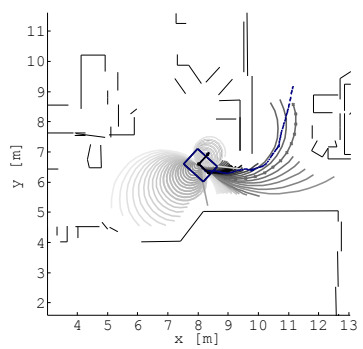
whereas the ground truth corresponds to intermediate curvatures to the left, cf. figure 4.28(b). Intermediate curvatures to the left, both forwards and backwards, remain likely in figure 4.28(c). This is due to a gradual decrease of mean linear velocity for left-turning user plans. The user's linear input is low enough to allow also backwards leading trajectories in the user model. The posterior in figure 4.28(d) represents another example of spatio-temporal filtering. Backwards leading plans as well as forward trajectories deviating from ground-truth have been either eliminated or their probability reduced. In this situation the posterior is more certain and resembles ground-truth closer.



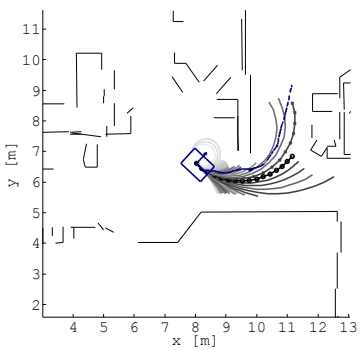
(a) all trajectories



(b) probabilities



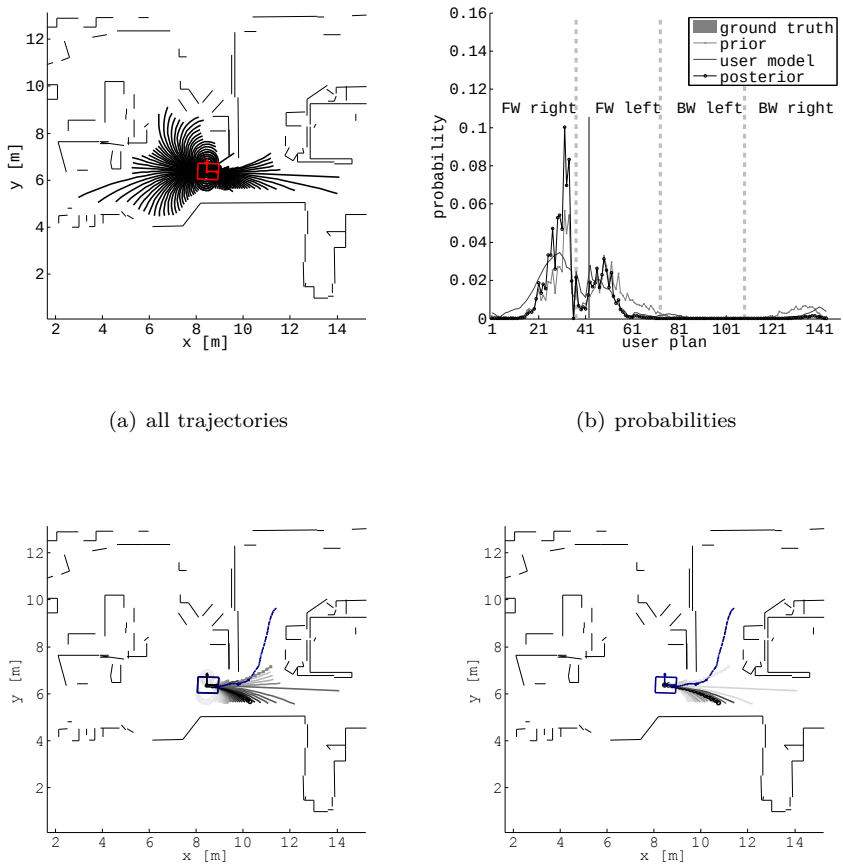
(c) user model



(d) posterior

FIGURE 4.26: User requesting left turn of intermediate curvature at time step 265.

Figure 4.29 shows the user decelerating before turning. In this case she is coming very close to the wall and slowing down is necessary. A very low linear velocity input of just $0.05m/s$ creates a bi-modal user model with maxima around left turning arcs of intermediate curvature leading forwards and backwards (please refer to figure 4.29(c)). The most probable posterior trajectories still point forwards in figure 4.29(d), in coherence with past driving. Figure 4.29(b) retraces how the posterior results from a prior with a peak at right turns of low curvature and the bi-modal user model. The prior reflects that the wheelchair is still turning right at time step 287 and advancing forwards.



(a) all trajectories

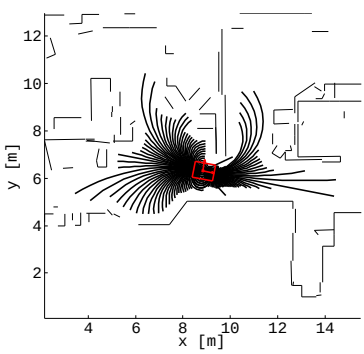
(b) probabilities

(c) user model

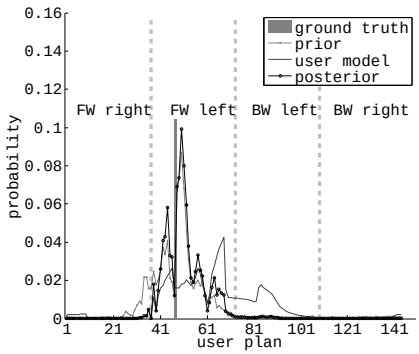
(d) posterior

FIGURE 4.27: Multi-modal user model at time step 275.

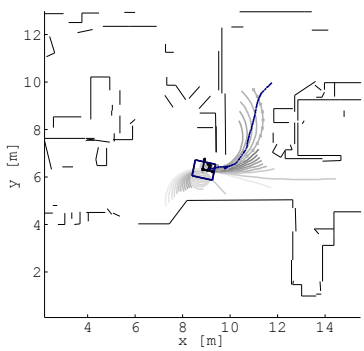
Another interesting situation occurs at time step 291 represented in figure 4.30. The user still requests a left turn, but pointing the joystick backwards ($\boldsymbol{u} = (v, \omega) = (-0.13m/s, 0.16rad/s)$). Please note that a positive angular velocity and negative linear velocity causes the wheelchair to turn counter-clockwise or to the left while advancing backwards (cf. the trajectory legend in figure 4.20). Figure 4.30(c) displays the very uncertain user model of time step 291. The largest part of the likelihood spreads almost uniformly over backwards arcs, both left and right turning, with a maximum around left-arcs of high curvature (figure 4.30(b)). Concerning forward arcs, right and left turns of large curvature are likely. We can link the high



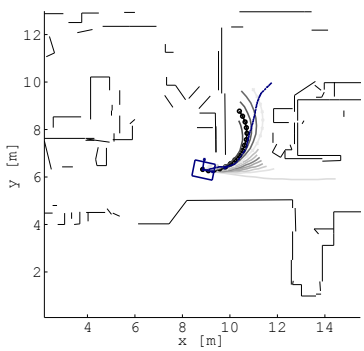
(a) all trajectories



(b) probabilities



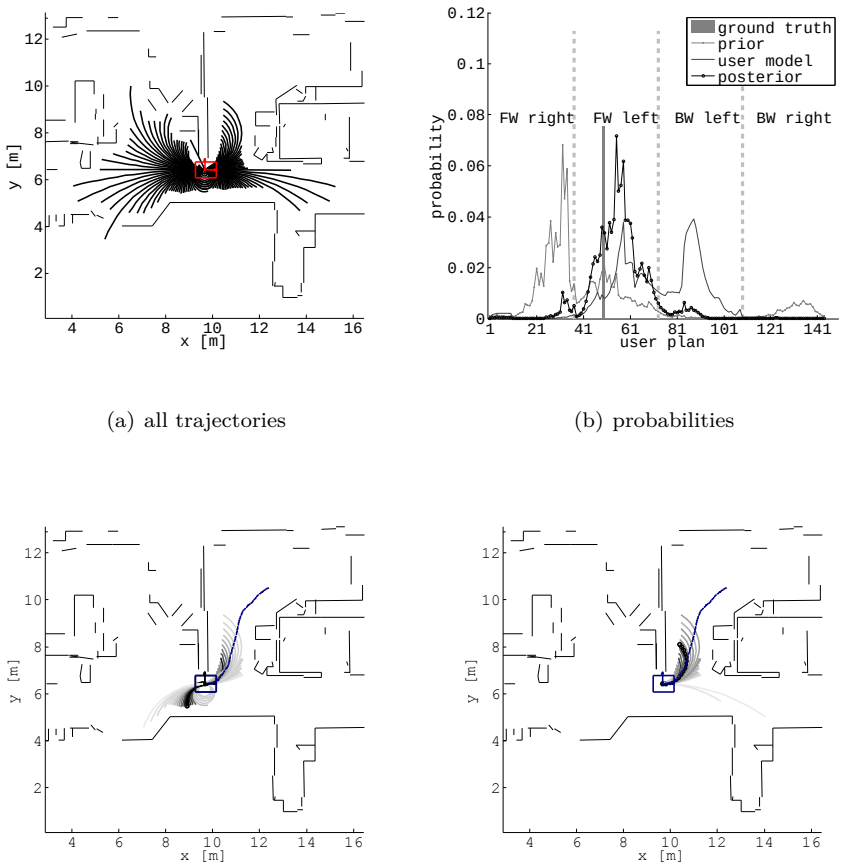
(c) user model



(d) posterior

FIGURE 4.28: Left-turn at time step 279.

uncertainty in the user model to the proximity of the user's input to the velocity origin. The posterior in figure 4.30(d) fails to eliminate the user model's uncertainty. A rather uniform prior in plot 4.30(b) transfers most of the user model's likelihood to the posterior. This situation is one of the few marked outliers in the posterior probability plot (cf. figure 4.24) in relation to the ground-truth hypothesis. At time step 291, the user is deviating considerably from her long-term goal of reaching the kitchen in the upper right map corner because of the proximity to a wall. Locally, her primary goal is staying away from the wall. Further, the posterior fails to filter the user model according to previous driving and both distributions resemble each



(a) all trajectories

(b) probabilities

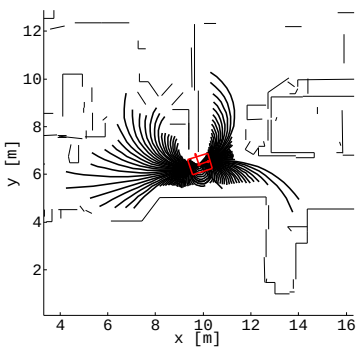
(c) user model

(d) posterior

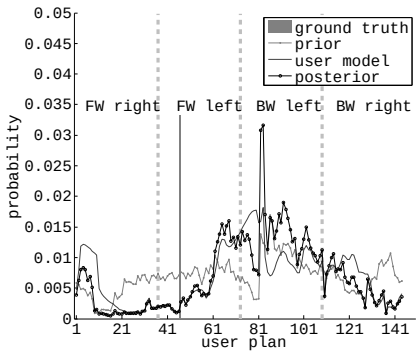
FIGURE 4.29: User requesting left-turn at time step 287.

other.

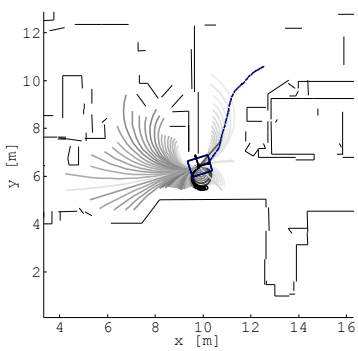
In figure 4.31 (situation 297) the user commands the wheelchair to advance forwards while turning to the left ($\boldsymbol{u} = (v_{user}, \omega_{user}) = (0.56m/s, 0.12rad/s)$). Given the low angular velocity input, forward arcs turning left and right are likely in the user model of figure 4.31(c). Interestingly, the user model is bi-modal and the forward straight path has a lower likelihood than any of two modes shown in figure 4.31(b). Forward paths of low or zero curvature are considerably shorter than neighbouring paths of slightly higher curvature. Bi-modality originates in



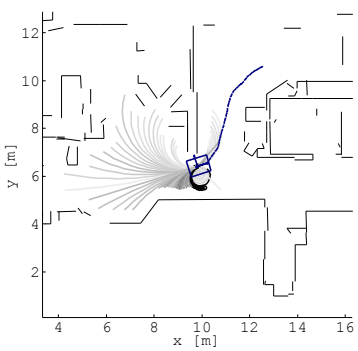
(a) all trajectories



(b) probabilities



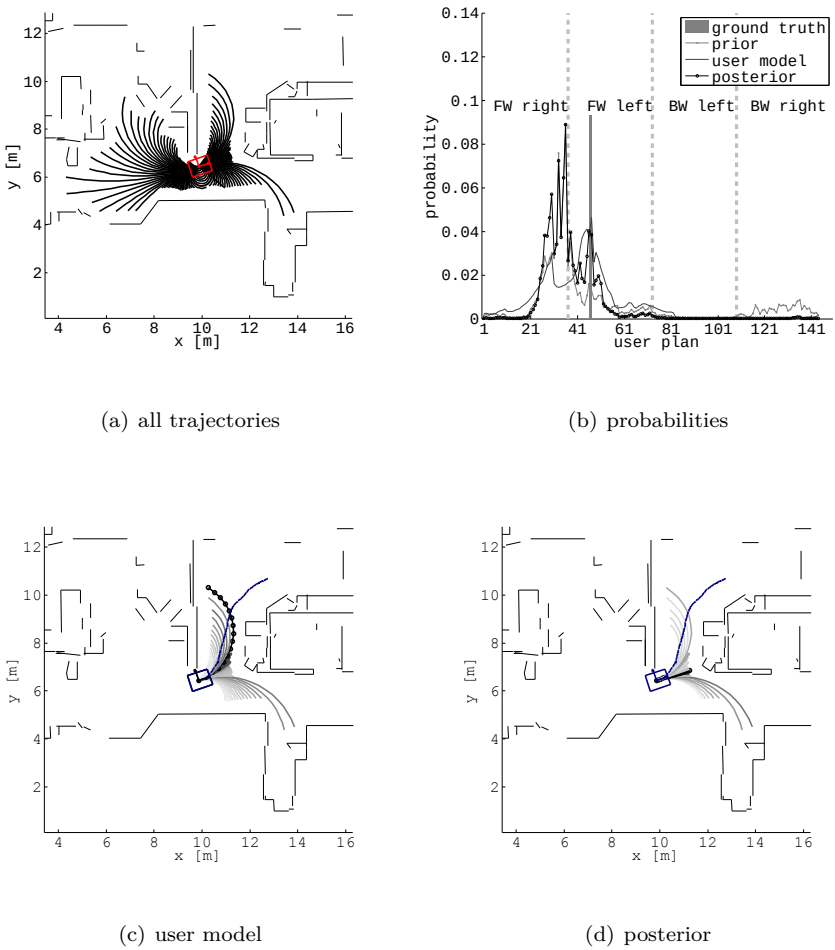
(c) user model



(d) posterior

FIGURE 4.30: User requesting backwards left turn at time step 291.

this case from the user model's preference for longer paths. Shorter paths can only be tracked at lower velocity than the one requested by the user. The most likely user plan is a counter-clockwise (left) turning arc of low-to-intermediate curvature. However, at time step 297 the wheelchair is turning right at low angular velocity. Therefore, the most probable posterior plans are right turns of low curvature (compare figure 4.31(d)). Still, left turning arcs, which locally represent ground-truth more accurately, remain half as probable as right turns. The posterior preserves the bi-modality of the user model as can be appreciated in the probability plot of figure 4.31(b). This information is potentially valuable to a



(a) all trajectories

(b) probabilities

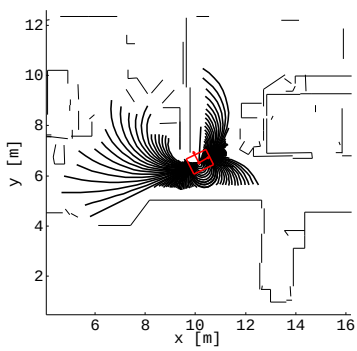
(c) user model

(d) posterior

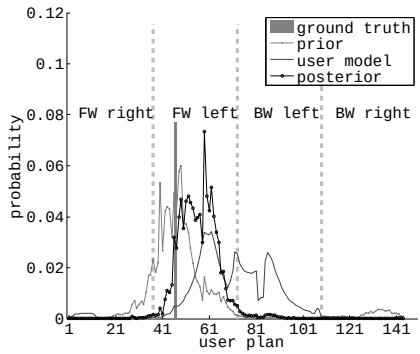
FIGURE 4.31: Bi-modal posterior at time step 297. The user is requesting a left turn of low curvature while the wheelchair is still turning right.

shared control algorithm. Instead of overcommitting to a left or right-turn, the algorithm could wait until more information becomes available before taking an action. Alternatively, it could decide to choose actions that disambiguate between left and right turns. The shared control algorithm has the possibility to select among exploration or exploitation strategies, with the goal of maximising driving comfort.

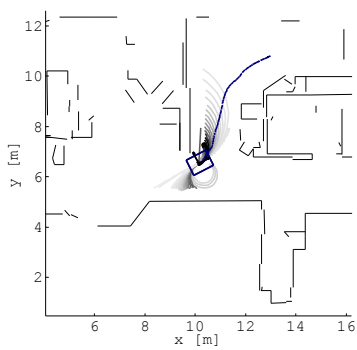
At time step 300, cf. figure 4.32, the user is requesting a high curvature turn



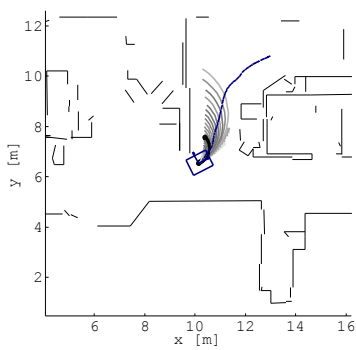
(a) all trajectories



(b) probabilities



(c) user model

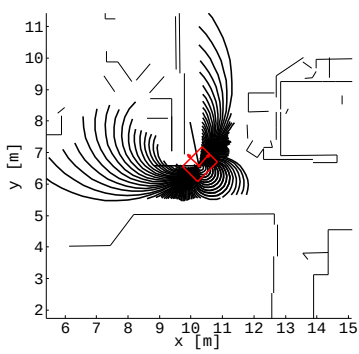


(d) posterior

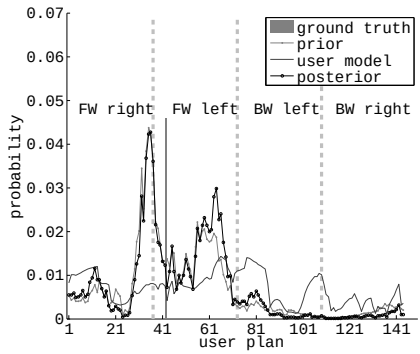
FIGURE 4.32: User requesting a high curvature left-turn at time step 300.

left. In the spatial user-model plot, figure 4.32(c), the most likely trajectories comprise forwards and backwards left-turning arcs ranging from intermediate to high curvature. Although the maximum likelihood plan points forwards, backwards plans are also likely due to the low linear input of merely $0.04m/s$. The posterior plot, figure 4.32(d), indicates a preference for medium curvature left turns, which represents the combination of past driving behaviour with the current user model.

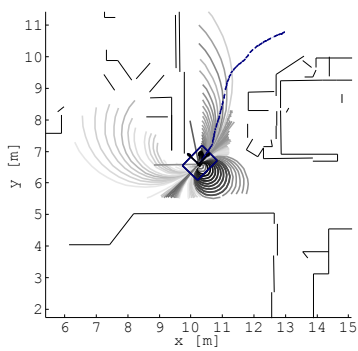
Situation 303 in figure 4.33 shows a highly uncertain user model. Probability mass spreads almost uniformly over many plans in the spatial user model plot of



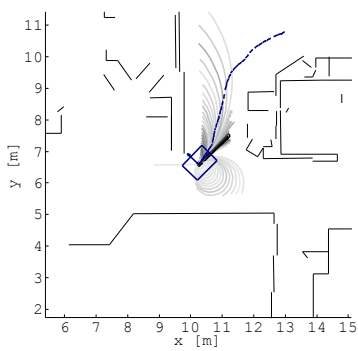
(a) all trajectories



(b) probabilities



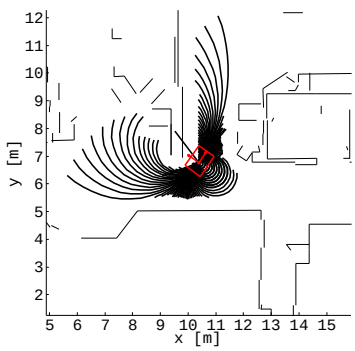
(c) user model



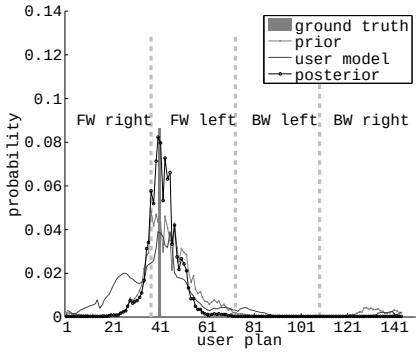
(d) posterior

FIGURE 4.33: A highly uncertain user model at time step 303.

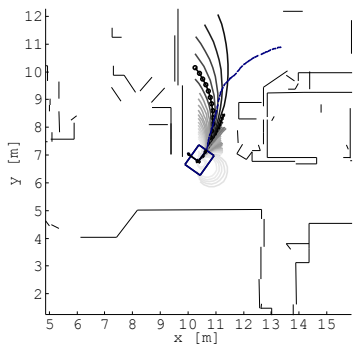
figure 4.33(c). A slight bias appears towards high curvatures left and right for forwards and backwards plans. Left turns are uniformly likely and some residual user likelihood appears at low curvatures of plans advancing backwards. The high uncertainty is linked to the proximity of the user's input to the velocity origin, ($\boldsymbol{u} = (v_{user}, \omega_{user}) = (0.11m/s, 0.12rad/s)$). In the posterior, figure 4.33(d), only forward advancing trajectories remain probable. According to the probability plot of figure 4.33(b), the straight plan advancing forward and low curvature right turns are the most probable a posteriori, followed by left and right turns of intermediate curvature.



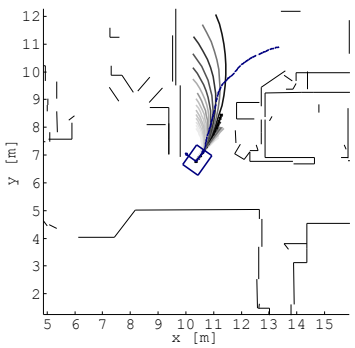
(a) all trajectories



(b) probabilities



(c) user model

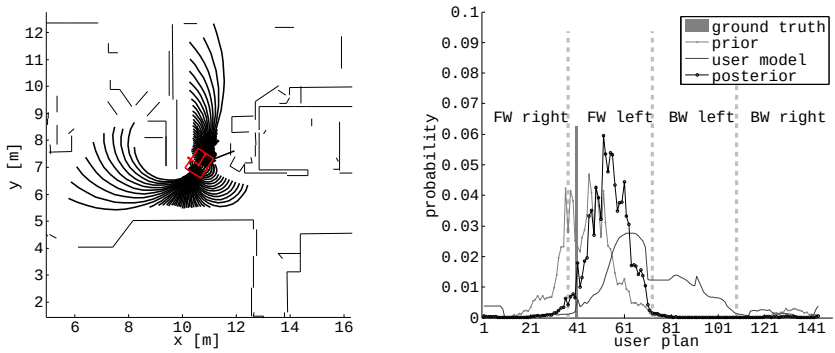


(d) posterior

FIGURE 4.34: Bias of the user model towards longer trajectories when user is requesting straight trajectory at time step 308.

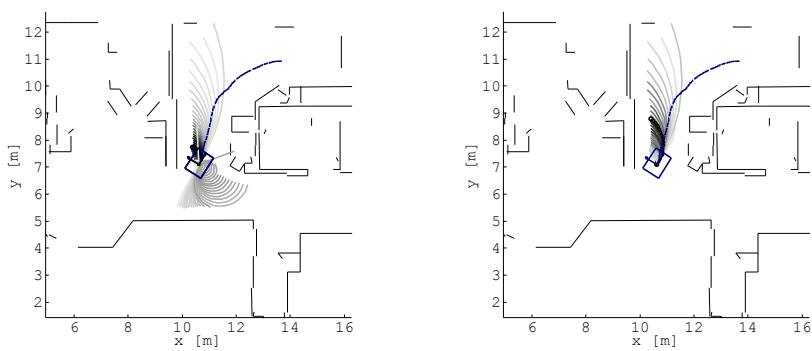
Figure 4.34 illustrates another interesting property of the user model. At time step 308 the deflection of the joystick is maximal for the linear velocity and zero for the angular input. Nevertheless, the maximum likely trajectory in the user model, figure 4.34(c), corresponds to a left-turning arc of low curvature. Further, the forward straight line reaches only half the maximal likelihood. The likelihood decreases gradually along right turning arcs on one side, and left turning arcs of higher curvature on the other side of the mode in the probability plot (figure 4.34(b)). This slight bias in the user model can be attributed to the trained

model's preference for longer trajectories, when the curvature is small. The bias originates in the angular component of the user model, $p_{user}\left(\omega_u(t) \mid \boldsymbol{i}_k\right)$, while estimating the zero-crossing angular velocity. A posteriori, figure 4.34(d), only left turning arcs of medium to low curvature and the straight line remain probable. This is the result of merging past driving -the wheelchair is still turning left at time-step 308- with the biased user model.



(a) all trajectories

(b) probabilities



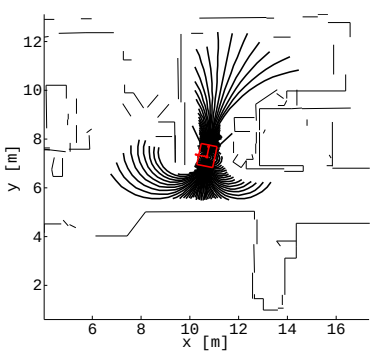
(c) user model

(d) posterior

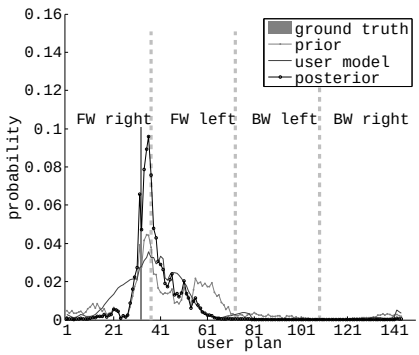
FIGURE 4.35: Left turn at high angular velocity during time step 312.

The following significant situation is time step 312, where the user requests a left turn at high angular speed. Frequently, for such turns the user prefers low linear speeds in order to control the wheelchair better. The likely trajectories in the user

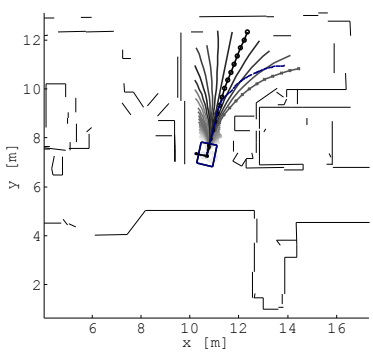
model for situation 312, figure 4.35(c), comprise both forward and backward leading arcs as the user's linear input is close to zero. The most probable trajectories a posteriori in figure 4.35(d) are reduced to forward left-turning arcs of intermediate curvature. These curvatures represent the overlap between prior and user model, which we show in figure 4.35(b). Past driving behaviour introduces inertia in the change of the most probable user plan and the posterior only gradually reflects the user model's information.



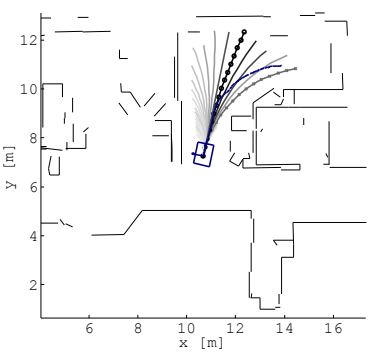
(a) all trajectories



(b) probabilities



(c) user model

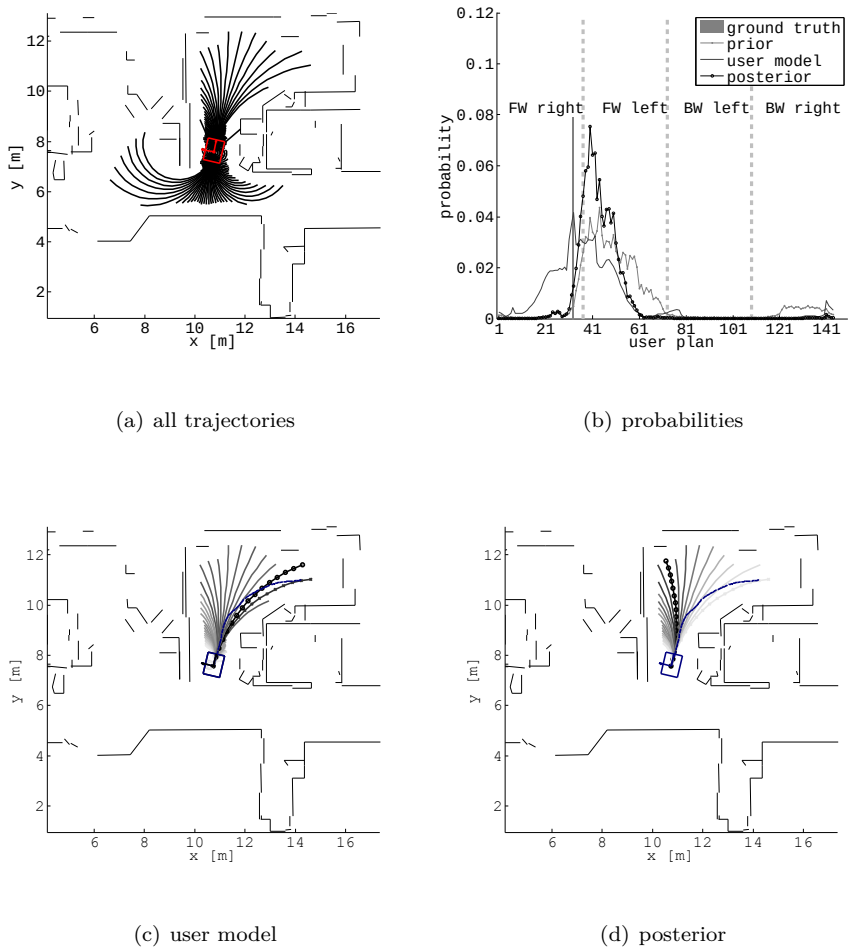


(d) posterior

FIGURE 4.36: User requesting trajectory straight-ahead, which results in an unbiased user model at time step 316.

Time step 316 proves the preference in the user model for longer trajectories.

Contrary to time-step 308, shown in figure 4.34, the most likely path in figure 4.36(c) has almost zero curvature. In both cases, the user input conveys to advance straight-ahead at full speed without turning. However, for both situations user plans differ in collision-free trajectory lengths. The probability plot for the user model (figure 4.36(b)) decreases along left and right arcs of low to intermediate curvature, given the zero angular input. The posterior in figure 4.36(d) is fairly peaked around the straight line and low curvatures left and right. The tail for left curvatures is longer, as the wheelchair is turning left with an intermediate curvature. Hence, the prior in figure 4.36(b) assigns large probability mass to such left turns.



(a) all trajectories

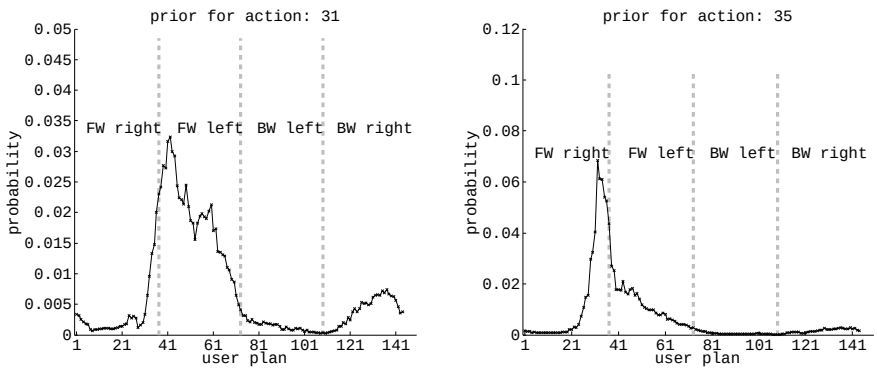
(b) probabilities

(c) user model

(d) posterior

FIGURE 4.37: Right turn with imprecise posterior at time step 320.

Figure 4.37 of situation 320 shows a snapshot of the user starting gradually to turn right. The user's linear input is almost maximal velocity whereas the angular input is only slightly negative. Hence, in the user model, figure 4.37(c), right turns of low curvature as well as the straight path are the most likely. Since the user's angular input is only slightly negative, arcs turning left are also probable according to the user model. A posteriori, user plans turning left and the straight trajectory have the highest probability in figure 4.37(d). It is interesting to note that in this case the posterior is partially converging to wrong user plans regarding long-term driving behaviour. This stems from the fact that left-turning arcs maintain probability weight in the user model as well as from an imprecise plan process function.



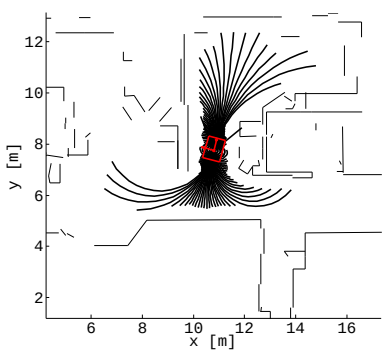
(a) action 31

(b) action 35

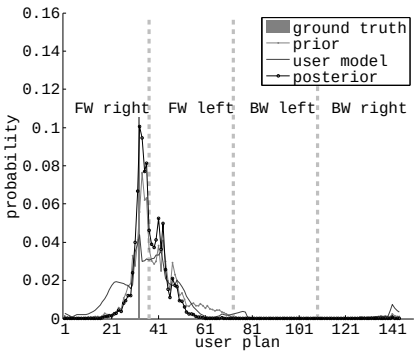
FIGURE 4.38: Action prior for two right turns when starting from a uniform distribution.

The prior corresponding to the right turn at the wheelchair's velocity at time step 320 is shown in figure 4.38(a). This plot displays the prior after executing the action at time step 320 if the initial distribution is uniform. The highest peak in the prior for this action involves a left turn instead of a right turn. Imprecisions in the plan process function and the broad user model likelihood cause the posterior to favour wrong plans in time step 320 (figure 4.37). In figure 4.37(d) convergence towards the wrong plan has to be understood in terms of long-term behaviour. For such cases, an overconfident posterior would be undesirable due to the convergence mistake. Wrong convergence is also named mode confusion in literature, which arises if the robot wrongly estimates the user's plan. By design, in order to mitigate the effects of mode confusion, a plan process function was selected with a considerable level of noise.

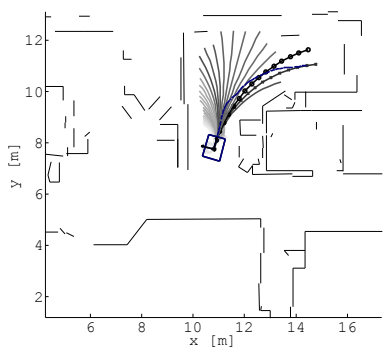
A few time steps later, situation 322 in figure 4.39, the posterior correctly combines the user model information with a proper prior for right turns (figure 4.38(b)).



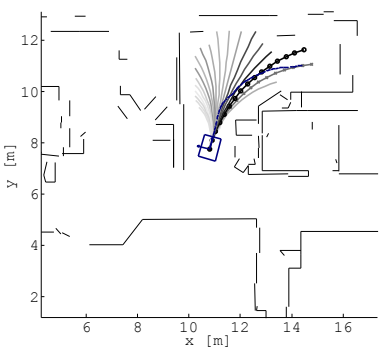
(a) all trajectories



(b) probabilities



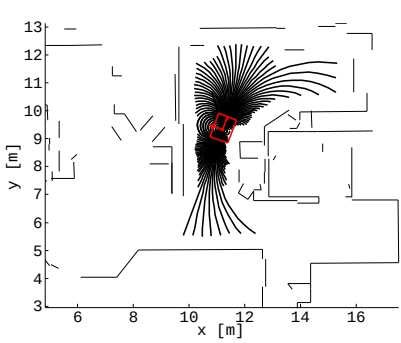
(c) user model



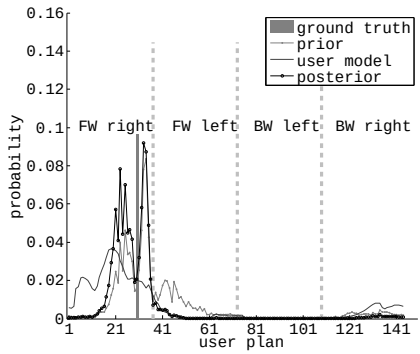
(d) posterior

FIGURE 4.39: Right turn with more precise posterior at time step 322.

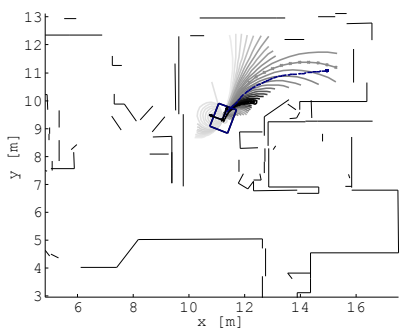
The user model (figure 4.39(c)) shows a slight preference for right turns of low curvature. Left turning arcs are likely as well. This is consistent with an input of high linear velocity and negative angular velocity of small absolute value. In the posterior, figure 4.39(d), the most probable trajectories turn right, which matches ground-truth. Considerable probability is assigned a posteriori to the straight line and left turning arcs of low to intermediate curvature, cf. figure 4.39(b). Avoiding an overconfident posterior causes that probability mass of the broad user model spreads a posteriori to plans deviating from ground-truth.



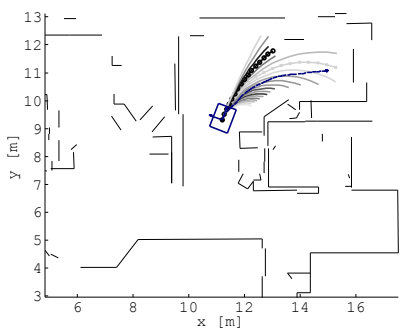
(a) all trajectories



(b) probabilities



(c) user model

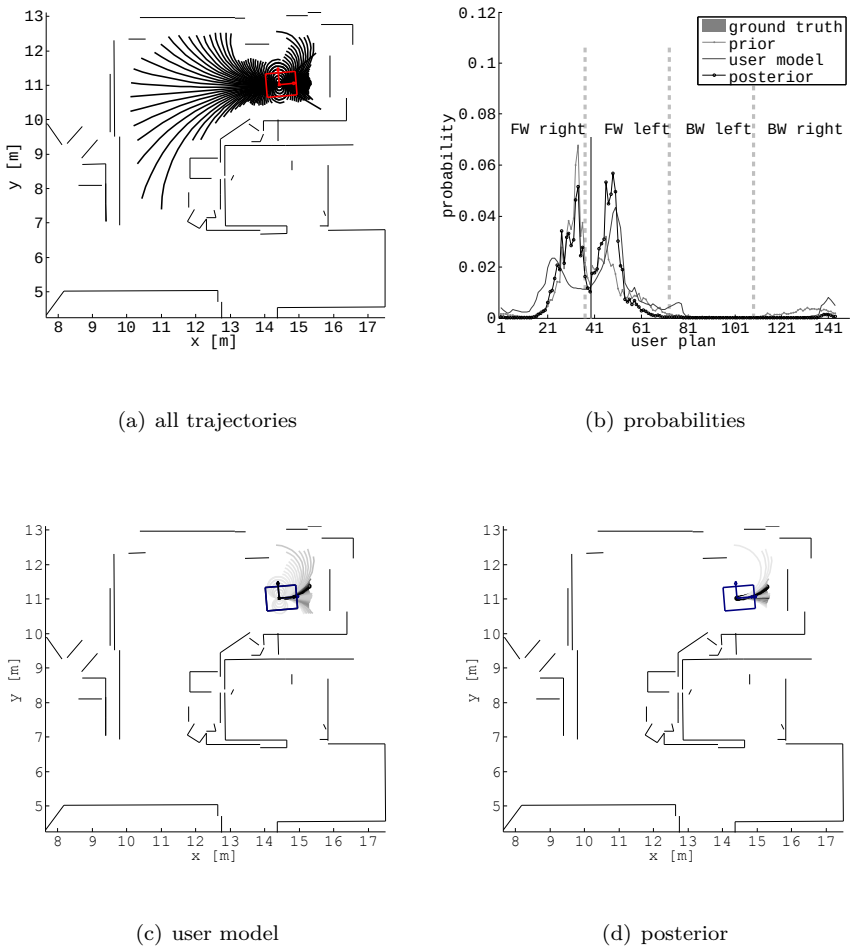


(d) posterior

FIGURE 4.40: User requesting higher curvature than necessary to follow future path at time step 332.

Time step 332 provides another example of where the user's short-term driving deviates from her long-term intention. The user is requesting a higher curvature than needed to follow the future path. In the probability plot, figure 4.40(b), much of the user model's likelihood accumulates around high and intermediate curvatures to the right. A smaller lobe of the user model covers low right curvatures. As can be appreciated in the spatial user model of figure 4.40(c), backwards turns of high curvature remain likely, albeit with less than one fourth of the maximal likelihood.

The residual likelihood originates from the relative proximity in velocity space[9] of the user's linear input to the estimated means of right-turning backwards-advancing plans. These mean linear velocities are low in absolute value, which lead to high curvatures. For instance, the mode in the user model at backwards right-turns originates from an estimated mean linear velocity within two standard deviations of the user's linear input. The posterior in figure 4.40(d) is centred on intermediate and low-curvature right turns, which resemble ground-truth closer.



(a) all trajectories

(b) probabilities

(c) user model

(d) posterior

FIGURE 4.41: End of manoeuvre at time step 355.

[9]Here proximity should be understood as metric proximity according to the Euclidean distance.

Figure 4.41 presents the final stage of the first manoeuvre at time step 355. The user's linear input is still high while the angular input is only slightly positive. In the probability plot, figure 4.41(b), we observe that the user model has two modes with a clear maximum at left arcs. Surprisingly, the likelihood decreases for left and right turns of low curvature and for the straight line. This has to do with a drop of the estimated mean linear and angular velocities for paths of low curvatures compared to paths of intermediate curvatures. Low curvatures require longer trajectory lengths to reach higher predicted mean velocities in the user model. The posterior, figure 4.41(d), presents a similar but narrower distribution as the user model with peaks shifting towards lower curvatures.

Driving from the kitchen to the bathroom

This section analyses how the user reaches the bathroom from the kitchen. Her driving style resembles the previous manoeuvre in many situations. For instance, she tends to slow down when turning at high angular velocity. Therefore, the focus of the current section is on novel situations instead of on an extensive case-by-case analysis, which we reported previously.

The estimated robot poses for this manoeuvre are shown in figure 4.42 and the input of the user appears in figure 4.43. At each time step along the path the robot evaluates the user model, i.e. the likelihood of all user plans. This distribution is displayed in figure 4.44. Special mention deserve time steps 585 until 592, where the user model is uniform because of zero user input. In the distribution plot of figure 4.44 these situations appear as a uniform grey stripe.

From the user model and a plan prior the robot derives the posterior distribution shown in figure 4.45. The uniform grey stripe is shifted in time due to the wheelchair's inertia. Although the user is giving no input starting at time step 585, the wheelchair continues to move. Also, the movement's onset is delayed compared to the first non-zero input.

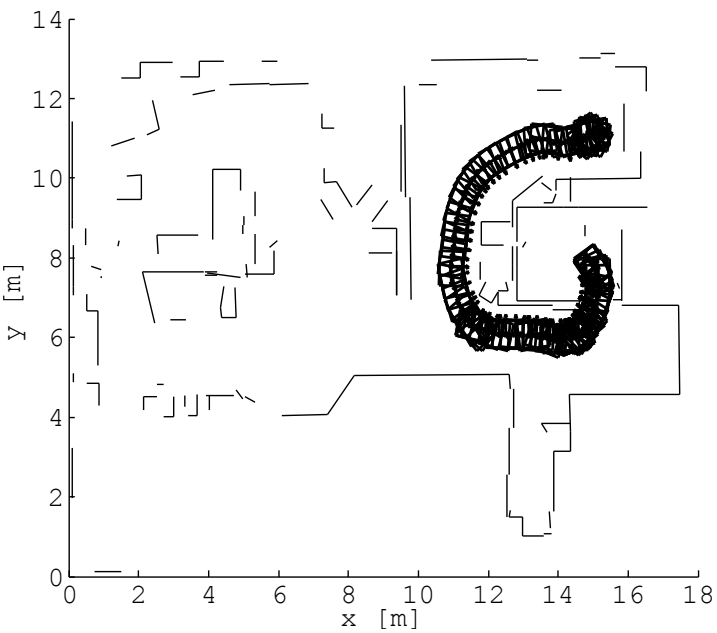


FIGURE 4.42: User's trajectory when driving from the kitchen to the bathroom.

The user initiates to move by turning on the spot to the right. Her linear velocity input has low absolute value and reverses its sign along the first time steps. Hence,

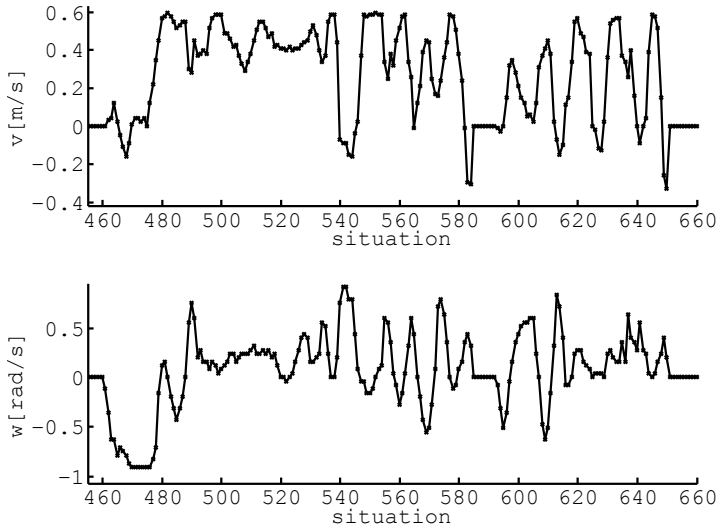


FIGURE 4.43: User's input while driving from the kitchen to the bathroom.

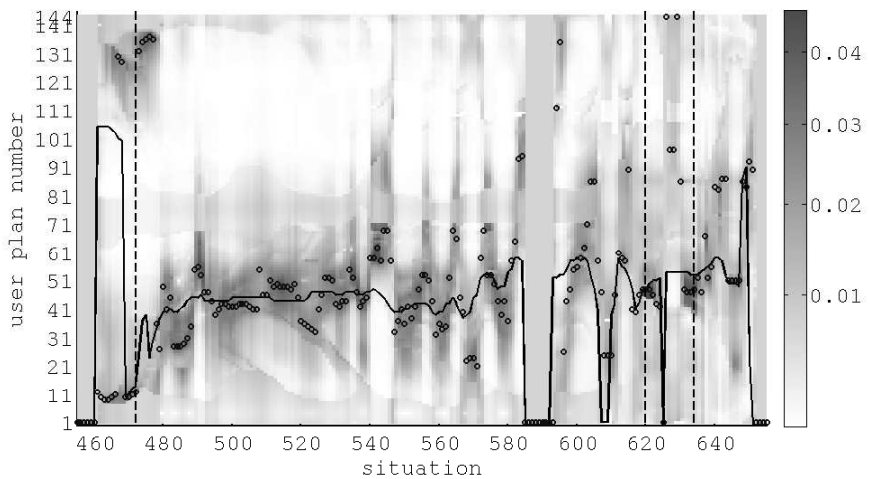


FIGURE 4.44: User model while driving from the kitchen to the bathroom.

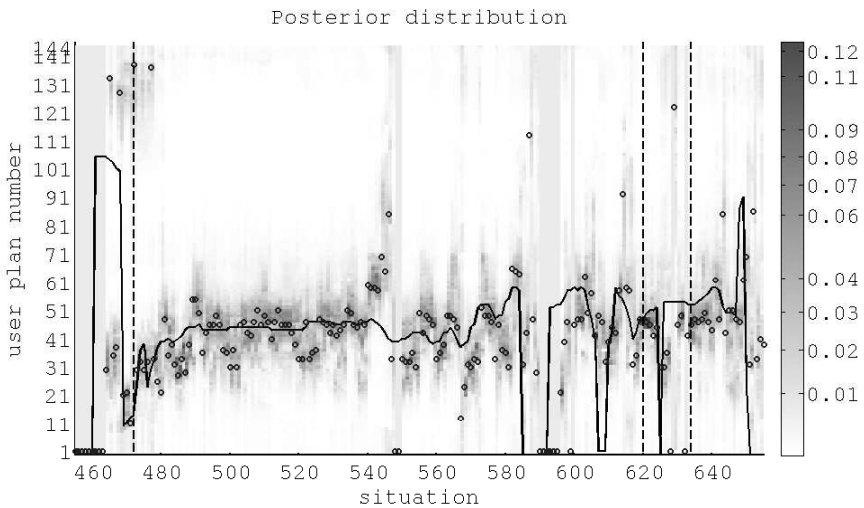
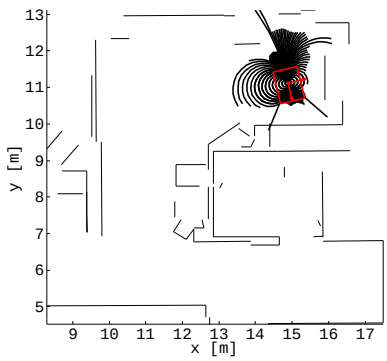


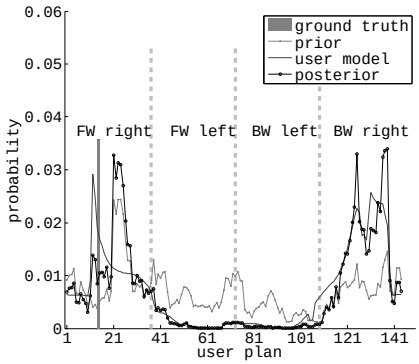
FIGURE 4.45: Posterior distribution over user plans while driving from the kitchen to the bathroom.

the user model, figure 4.44, and the posterior (figure 4.45) oscillate between forward and backward right turns at the movement's onset. A reference example for the start of the manoeuvre is shown in figure 4.46 and refers to situation 472. In the probability plot, figure 4.46(b), the user model distribution is bi-modal. Both forward and backward right-turns are likely with maxima at intermediate curvatures. The gradual decrease in likelihood along right-turns can also be appreciated in the spatial user model distribution of figure 4.46(c). Left turns of low curvature maintain some residual likelihood as the decrease rate of the user model is moderate. The posterior, according to the probability plot in figure 4.46(b), favours intermediate curvatures forwards and intermediate to high curvatures backwards. A posteriori (cf. figure 4.46(d)), the high curvature backwards user plan is a little more probable than the other hypotheses.

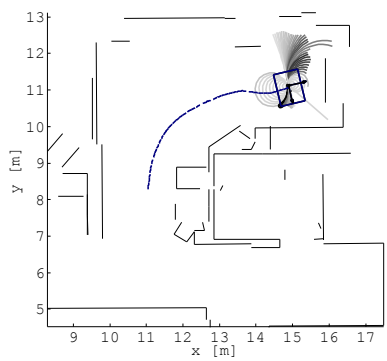
Time step 620, figure 4.47, shows the user entering the bathroom. The reduced space compared to the wheelchair's dimension leads to a fairly peaked user model at intermediate left curvatures. Only arcs entering the bathroom remain likely. The effect of the posterior is to reinforce the peak in the user model, which can be appreciated in the probability plot of figure 4.47(b). This leads to a very similar spatial distribution for the user model and posterior, shown in figure 4.47(c) and figure 4.47(d) respectively.



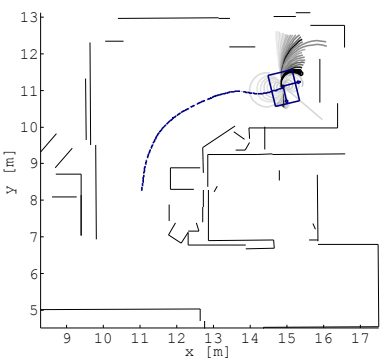
(a) all trajectories



(b) probabilities



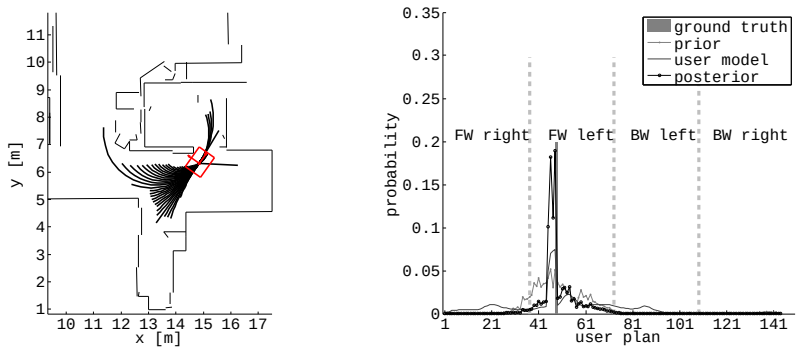
(c) user model



(d) posterior

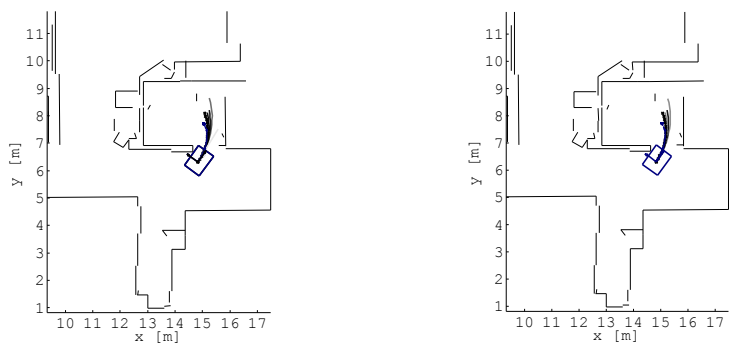
FIGURE 4.46: Start of the manoeuvre of figure 4.42 at time step 472.

A few time-steps later, when the user has already negotiated the entry to the bathroom, the amount of driving options widens. In situation 634, depicted in figure 4.48, a multitude of paths point straight-ahead and to the left. The user is requesting a high linear and a moderately positive angular velocity. This causes the user model, figure 4.48(c), to spread its likelihood around low and intermediate curvatures turning left. Compared to the previous situation in figure 4.47, more choices imply a broader user model distribution. The spatial plot of the posterior distribution, figure 4.48(d), presents a similar distribution to the user model but with lower uncertainty. The sharpening of the posterior after applying Bayes' rule



(a) all trajectories

(b) probabilities

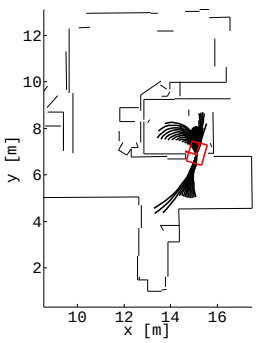


(c) user model

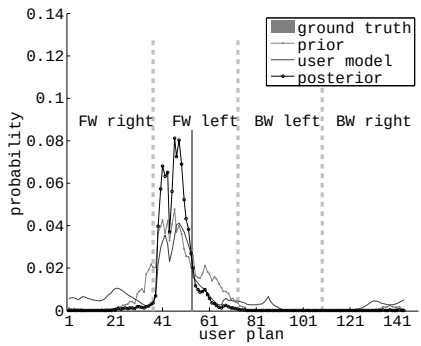
(d) posterior

FIGURE 4.47: Very peaked user model and posterior when entering the bathroom at time step 620.

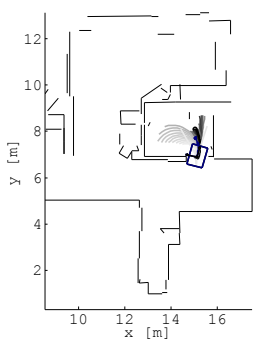
is shown in the probability plot of figure 4.48(b).



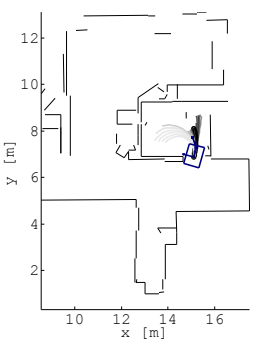
(a) all trajectories



(b) probabilities



(c) user model



(d) posterior

FIGURE 4.48: Broader user model and posterior once the wheelchair is within the doorframe at time step 634.

Leaving the bathroom and entering the small kitchen

This section analyses the trajectory of the user from the bathroom to the small kitchen. The complete manoeuvre is depicted in figure 4.49, while the input of the user appears in figure 4.50. Further, the robot's estimated user model is shown in figure 4.51. The posterior distribution in figure 4.52 represents the sequential combination of current driving expressed by the user model and past driving. Again, a uniform grey stripe interrupts the user model during time-steps 760 to 780. In this case, the user gives some initial input, but needs to re-evaluate her strategy to leave the bathroom without colliding.

In order to avoid repetition, we will describe only the most salient and novel situations.

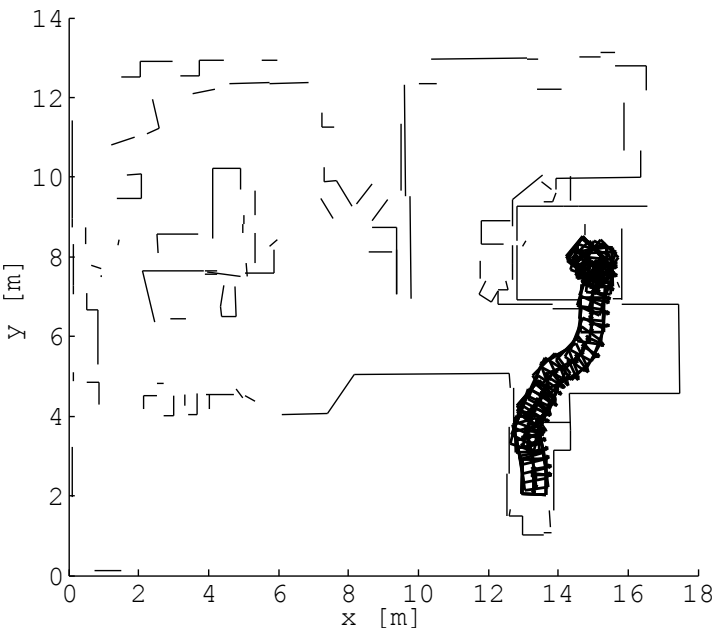


FIGURE 4.49: The user's trajectory when driving from the bathroom to the small kitchen.

The first interesting situation for the present manoeuvre occurs at the second movement attempt. After initially advancing forwards to achieve more manoeuvring space, then pausing shortly, the user starts turning almost on-the-spot while slowly moving backwards. Figure 4.53 shows a snapshot of this movement segment which refers to situation 778. The probability plot in figure 4.53(b) reveals a bi-modal user model distribution of left curvatures with maxima at forwards and backwards arcs. Interestingly, the likelihood drops for very high curvatures creating two distinguishable peaks. This effect originates in the angular component of the user

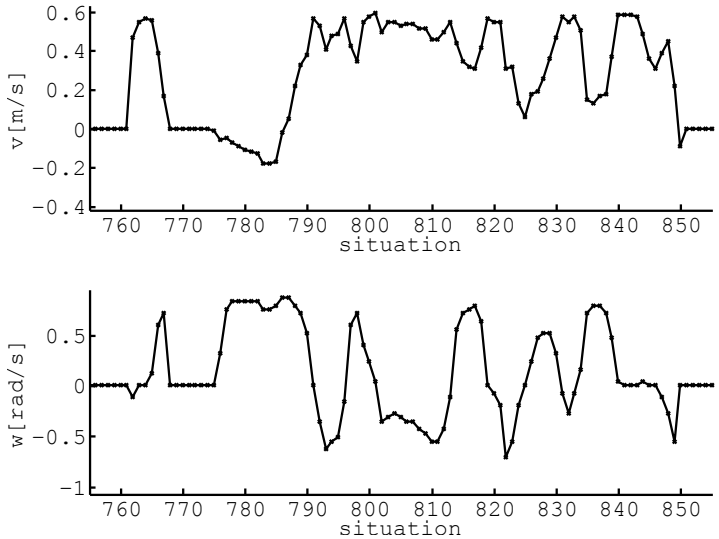


FIGURE 4.50: User's input while driving from the bathroom to the small kitchen.

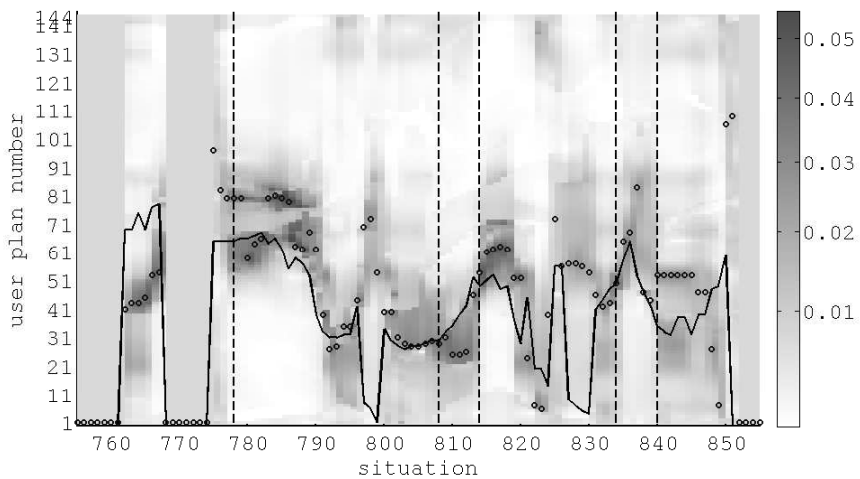


FIGURE 4.51: User model while driving from the bathroom to the small kitchen.

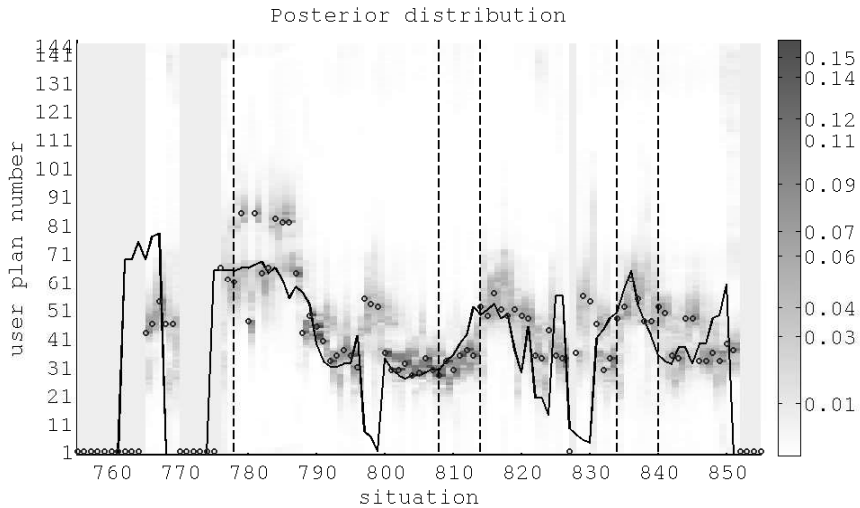
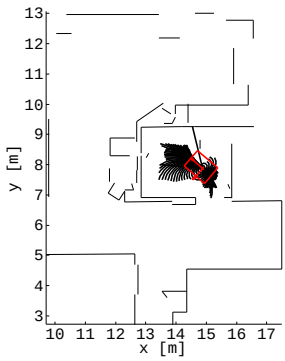


FIGURE 4.52: Posterior distribution over user plans while driving from the bathroom to the small kitchen.

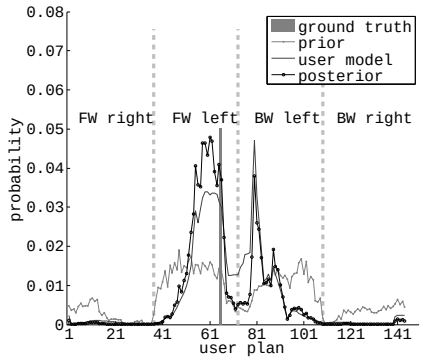
model, $p_{user}\left(\omega_k^u \mid \boldsymbol{i}_k\right)$, which is slightly biased towards intermediate trajectory lengths for high-curvature plan hypotheses. Remedying this bias would probably require more numerous training examples with high-curvature turns as ground-truth.

The posterior in situation 778 is only slightly more peaked than the user model due to the broad prior. The spatial user model in figure 4.53(c) is indeed very similar to the posterior of figure 4.53(d). However, the most likely user model plan points backwards whereas the most probable posterior plan indicates forward movement. The difference in probability between forwards and backwards modes remains small.

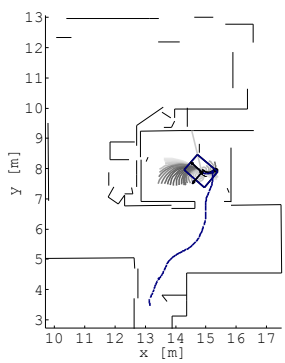
Situation 808, figure 4.54, shows the wheelchair after leaving the bathroom and reorientating in order to enter the small kitchen. The user model (figure 4.53(c)) assigns likelihood to a broad range of plan hypotheses. The likelihood's maximum occurs at intermediate right-turning curvatures and decreases along right turns of higher curvature on one side of the mode, and along left turns of low curvature on the other side. The likelihood's decrease is moderate, especially for right turns. In this case, the angular user model, $p_{user}\left(\omega_k^u \mid \boldsymbol{i}_k\right)$, presents similar mean angular velocities across curvatures. All likelihood variation for right turns originates in



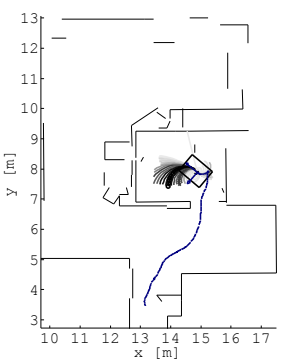
(a) all trajectories



(b) probabilities



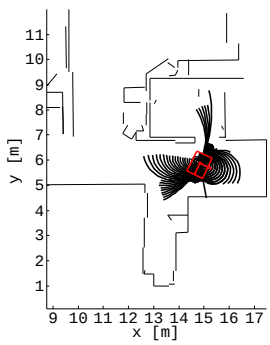
(c) user model



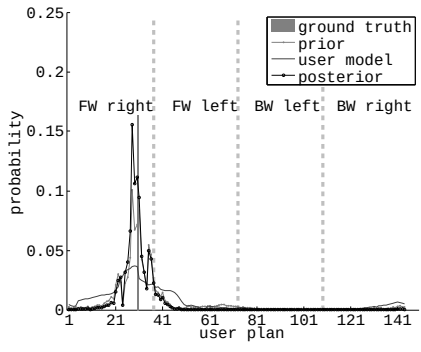
(d) posterior

FIGURE 4.53: Second attempt to initiate movement resulting in a bi-modal user model at time step 778.

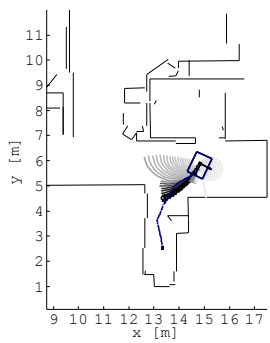
the linear velocity component, $p_{user}\left(v_k^u \mid \omega_k^u, \boldsymbol{i}_k\right)$. Left turns retain some likelihood due to an insufficiently rapid increase of the mean angular plan velocities. This introduces errors while estimating the zero-crossing of the mean angular plan velocities. The posterior in figure 4.54(d) considerably narrows down the set of probable hypotheses. The maximum a posteriori agrees with the user model's mode and provides a fair representation of ground-truth. Combining prior plan estimates with the current user model proves again very beneficial.



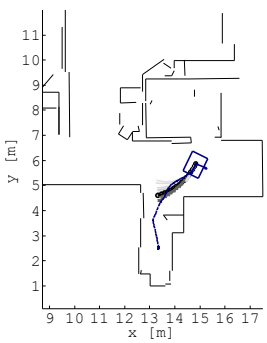
(a) all trajectories



(b) probabilities



(c) user model

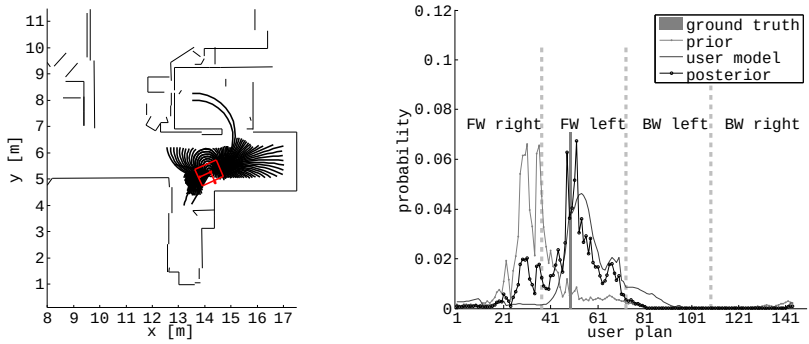


(d) posterior

FIGURE 4.54: The user turns right to enter the small kitchen resulting in a broad user model at time step 808.

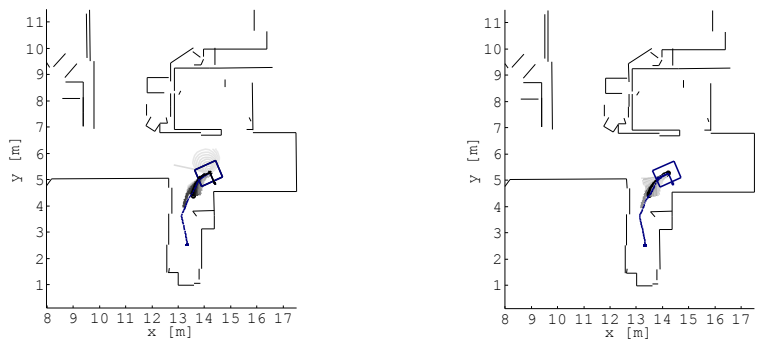
Figure 4.55 refers to situation 814, which is an inflection point in the current manoeuvre. The user points the joystick left while the wheelchair is still turning right at moderate angular speed. In the probability plot of figure 4.55(b), we observe how user model and prior disagree about the most likely user plan. The prior favours right turns, consistent with the wheelchair's prior movement, whereas the user model peaks around intermediate left curvatures. This discrepancy manifest itself in the spatial distribution of the posterior (figure 4.55(d)). A posteriori, a wider range of forward right and left turning arcs are probable than in the user

model of figure 4.55(c). However, the posterior completely eliminates backward left turns, which maintained marginal likelihood in the user model.



(a) all trajectories

(b) probabilities



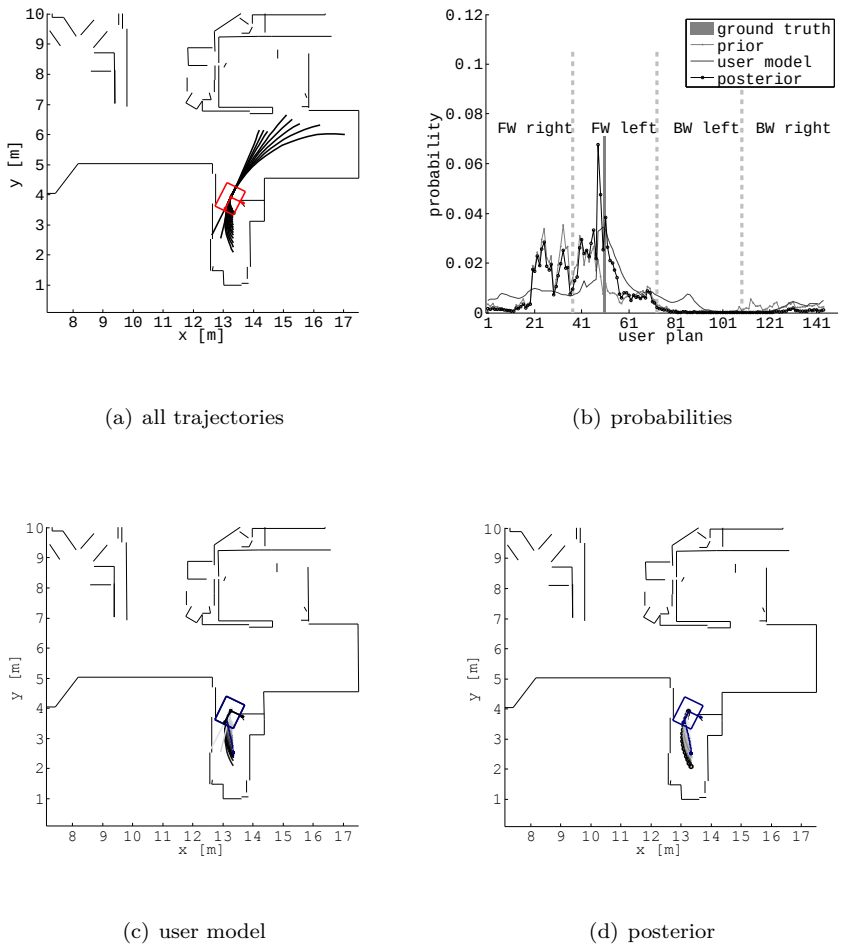
(c) user model

(d) posterior

FIGURE 4.55: Sudden change in user's plan leading to disagreeing user model and prior at time step 814.

Time step 834 shows the user in a difficult situation. At time step 834, the user can only turn left if she is to avoid a collision. The probability plot, figure 4.56(b), depicts a very narrow user model with a maximum around intermediate left curvatures and a peak-to-base-line ratio of around 3.5. In the spatial user model plot (figure 4.56(c)) most of the likelihood weight resides in left-turning arcs. The posterior preserves the user model's structure, despite a small reduction in

peak-to-base-line ratio. During the present situation the wheelchair is still turning opposite to the user's input and, thus, some probability mass leaks to right turning arcs. This discrepancy lasts only until the present time step. Starting from the next situation the wheelchair has changed course. In the spatial posterior plot, figure 4.56(d), the most probable hypothesis turns away from the wall.



(a) all trajectories

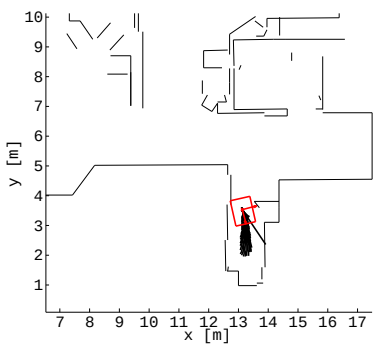
(b) probabilities

(c) user model

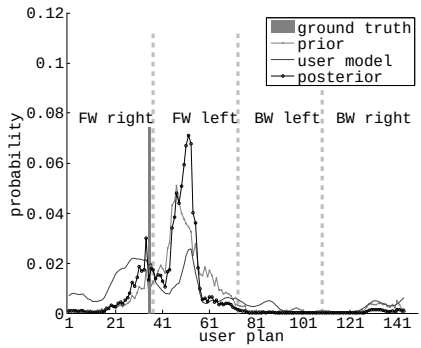
(d) posterior

FIGURE 4.56: User's predicament at time step 834.

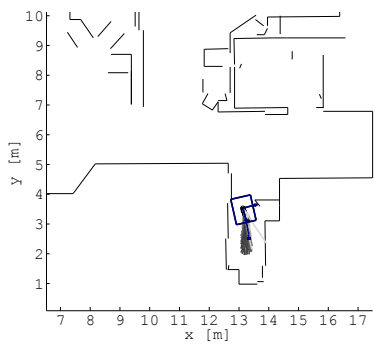
The next situation illustrates the final stretch of the manoeuvre to enter the small kitchen. In the complete user model plot, figure 4.51, this segment is characterised by a bi-modal distribution and covers situations 840 up to 850. From this set of situations, we have selected time stamp 840 in figure 4.57 as a representative



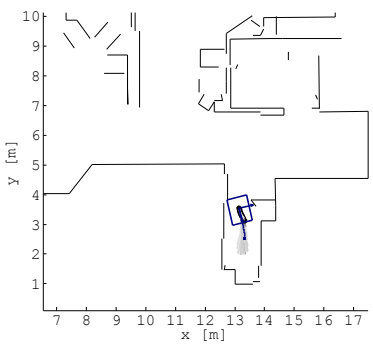
(a) all trajectories



(b) probabilities



(c) user model



(d) posterior

FIGURE 4.57: Bi-modal user model and posterior distributions at time step 840.

example. The user's joystick is maximally deflected regarding linear velocity and points slightly left. Accordingly, the user model assigns likelihood to right and left turns, which is displayed in the probability plot of figure 4.57(b). This plot also reveals that the user model drops at low left curvatures. In the spatial distribution of the user model, figure 4.57(c), one can observe how these curvatures have been shortened considerably compared to the previous situation because of the wheelchair's movement. The net effect translates into reduced mean linear velocities for each shortened plan, which in turn enhances the difference to the user's input. The large difference causes the likelihood to drop. At time step 840, the posterior

(figure 4.57(d)) converges to the user model's maximum at left curvatures because previously the user was turning left.

Leaving the small kitchen

This section analyses how the user leaves the small kitchen. The complete manoeuvre is depicted in figure 4.58 and the user's input in figure 4.59. The interest in this movement segment is twofold. Firstly, the user is mostly driving backwards and, secondly, there is very little manoeuvring space.

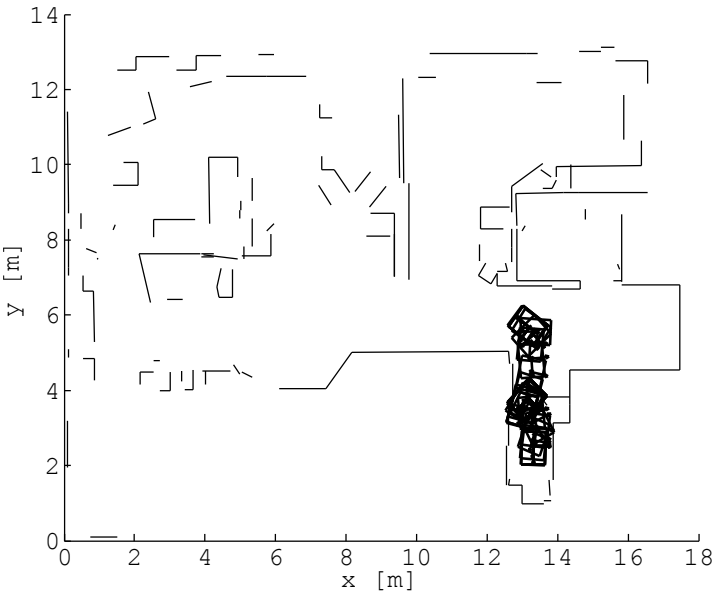


FIGURE 4.58: User's trajectory when leaving the small kitchen.

In the user model, figure 4.60, and posterior distribution, figure 4.61, the movement can be segmented into three separate clusters. Also the user's input in figure 4.59 shows three regions delimited by zero inputs. The initial part of the movement driving backwards concludes when the user collides with one of the walls. Then follows a long stretch of no input while the user is replanning her trajectory. The user has to drive forwards and turn left in order to re-orient the wheelchair to avoid colliding again. Afterwards, she exits the small kitchen driving backwards while turning clockwise (right). The third movement cluster concludes once the user reaches the corridor and the wheelchair faces the direction of the living room.

Figure 4.62 displays the start of the manoeuvre. The user has already given some initial input, but it is only from situation 985 onwards that the wheelchair moves appreciably. Most of the user model's likelihood in figure 4.62(b) and the spatial plot of figure 4.62(c) spreads over backwards arcs turning left and right. This is warranted by an angular velocity input of low magnitude. In the spatial plot of figure 4.62(d) it is possible to observe that the largest part of posterior probability concentrates around backwards turns right while some

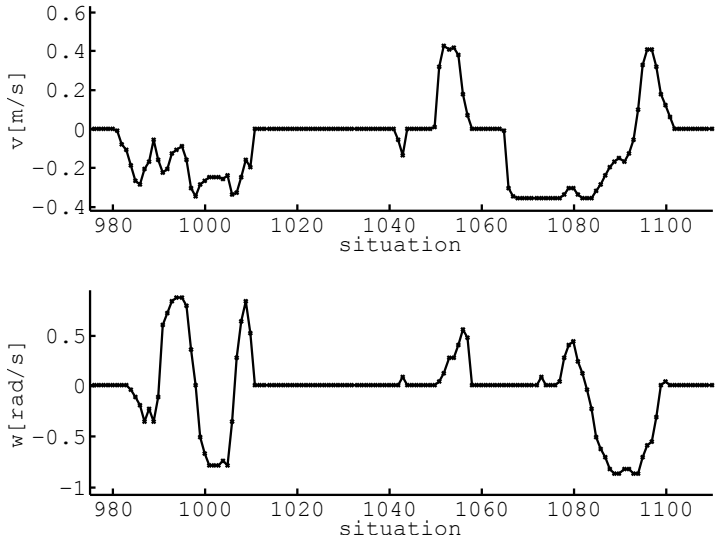


FIGURE 4.59: User's input while leaving the small kitchen.

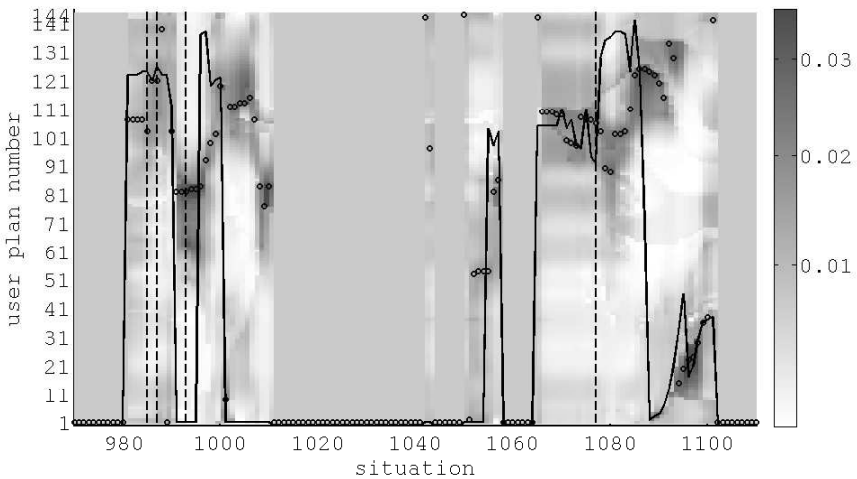


FIGURE 4.60: User model while leaving the small kitchen.

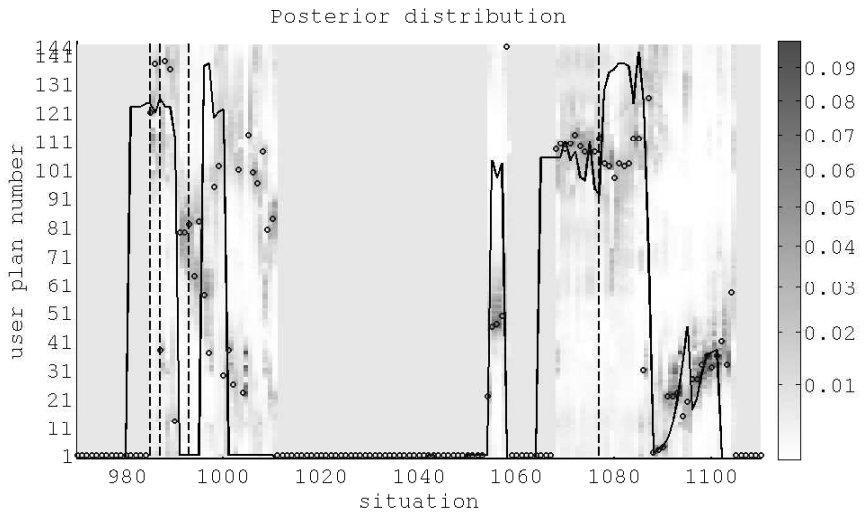
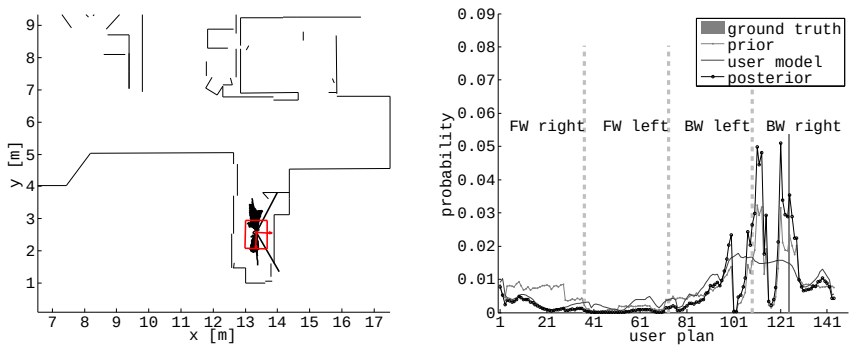


FIGURE 4.61: Posterior distribution over user plans while leaving the small kitchen.

residual probability remains at left turning arcs. Interestingly, in the graph of figure 4.62(b) the posterior presents three distinct modes; two at right turns and one lower at left backwards turns. Instead of decreasing smoothly like the user model, the posterior suddenly drops for mid-range backwards curvatures left and right. This discontinuity originates in the plan process function, which is more inaccurate for backwards actions due to a lack of training data. Inaccuracies in the plan process function appear frequently throughout the complete manoeuvre since the user is mostly driving backwards.

A further example of plan process function errors is shown in figure 4.63 and refers to time step 987. The user is pointing the joystick backwards and to the right. According to the user model in figure 4.63(c) the most likely plans lead backwards. In the probability graph of figure 4.63(b) the likelihood decreases from a maximum at low curvatures along right and left curvatures. Most likelihood gets assigned to right turns, where the likelihood diminishes only gradually reaching forward arcs of high curvature. Backwards leading plans preserve their probability a posteriori, but, as the probability graph (figure 4.63(b)) and the spatial plot (figure 4.63(d)) reveal, the plan process function shifts considerable probability to forward advancing arcs of low curvature. The maximum a posteriori points forwards even if the current action as well as the user's input require backwards movement. More training examples of backwards movement or a less sensitive plan process function would



(a) all trajectories

(b) probabilities

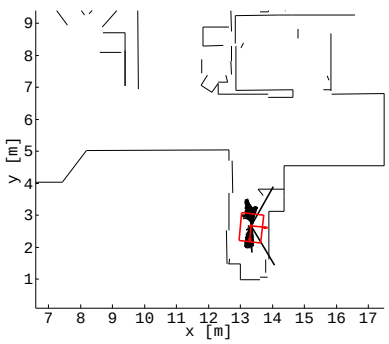
(c) user model

(d) posterior

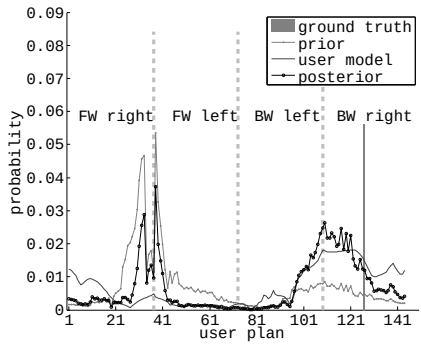
FIGURE 4.62: Initiation of backwards movement at time step 985.

alleviate such estimation errors.

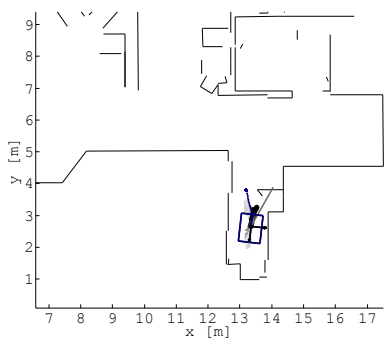
Plan process function inaccuracies are less damaging when there is absence of prior information such as in figure 4.64. At time step 993 the user is requesting a sharp left turn at low backwards linear velocity. High-curvature turns both forwards and backwards are very likely in the user model, with a slight preference for backwards arcs. The probability graph of figure 4.64(b) also shows a uniform prior across user plans. Such prior distribution results in an almost identical user model (figure 4.64(c)) and posterior (figure 4.64(d)).



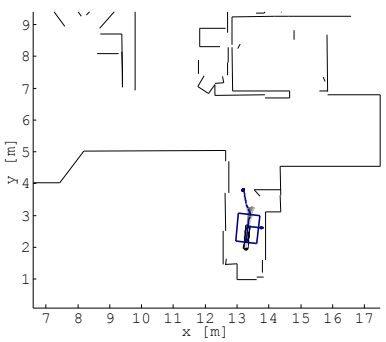
(a) all trajectories



(b) probabilities



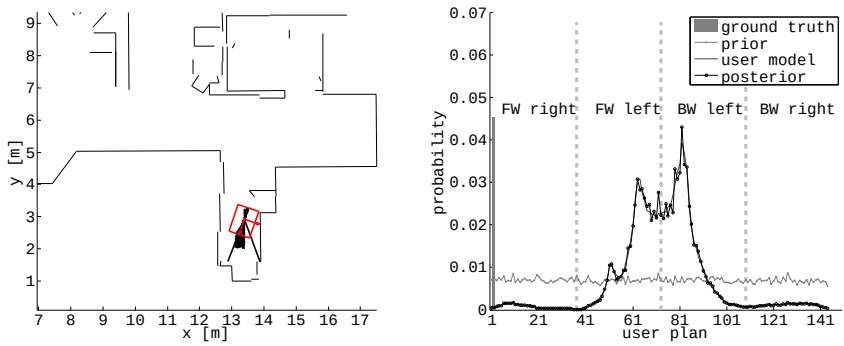
(c) user model



(d) posterior

FIGURE 4.63: Inaccurate plan process function while driving backwards at time step 987.

The remainder of the movement segment covering time steps 980 to 1010 in the user model of figure 4.60 deals with situations prior to the collision. In this part of the manoeuvre short trajectory lengths and inaccurate ground-truth estimation explain the large uncertainty of the user model as well as the divergence between ground-truth and user model. Future work will analyse how to improve ground-truth estimation at the end of a manoeuvre, because normally the last part of the manoeuvre entails short paths whereas the plan library can include longer trajectories reaching further ahead.



(a) all trajectories

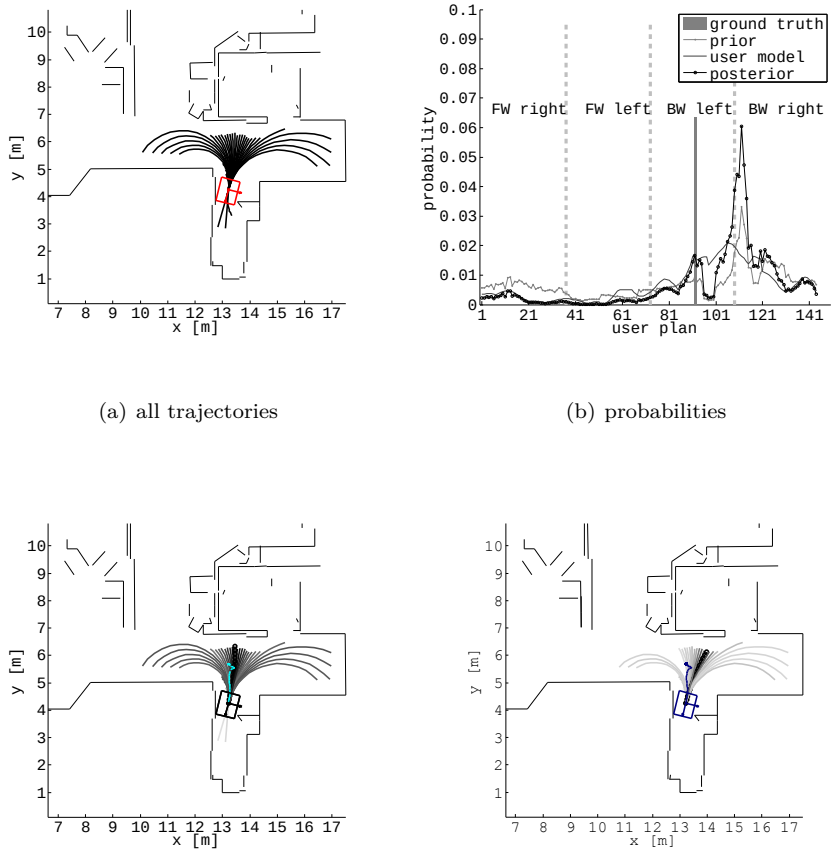
(b) probabilities

(c) user model

(d) posterior

FIGURE 4.64: Broad prior while turning sharply left at time step 993.

Figure 4.65 of time step 1077 belongs to the third data cluster of the current manoeuvre. After correcting the wheelchair's orientation, the user manages to exit the small kitchen. This snapshot illustrates the difficulty in estimating ground-truth at the movement's end. Collision-free backwards arcs reach further than the future path. In this case, ground-truth according to Hausdorff's metric corresponds to the rightmost arc viewed from the wheelchair's perspective. Clearly, other arcs would represent the user's intention more accurately than this path with little clearance of the wall. During time step 1077 the user is requesting a mid-range backwards linear velocity while slightly pointing the joystick left. The probability graph of



(a) all trajectories

(b) probabilities

(c) user model (future path in light blue)

(d) posterior

FIGURE 4.65: Driving backwards along a straight line at time step 1077.

figure 4.65(b) shows how the user model peaks at low-curvature backwards left turns and gradually decreases on both sides of the divide between left and right turns. In the spatial user model plot, figure 4.65(c), the complete range of available backwards arcs remains likely. The posterior, figure 4.65(d), is considerably more peaked around almost zero-curvature right turns, which match the wheelchair's prior movement. Still, small errors in the plan process function discard a reduced set of mid-range backwards left turns. Their posterior probability drops in the probability graph (figure 4.65(b)) and, consequently, a hole appears in the spatial posterior plot (figure 4.65(d)).

Driving from the corridor back to the living-room

This section describes the last part of the experiment, when the user is travelling from the corridor back to the living room. The complete trajectory appears in figure 4.66, and the user's input in figure 4.67. As with previous movement segments, we report the user model (figure 4.68), which combined with a movement prior, results in the posterior distribution of figure 4.69.

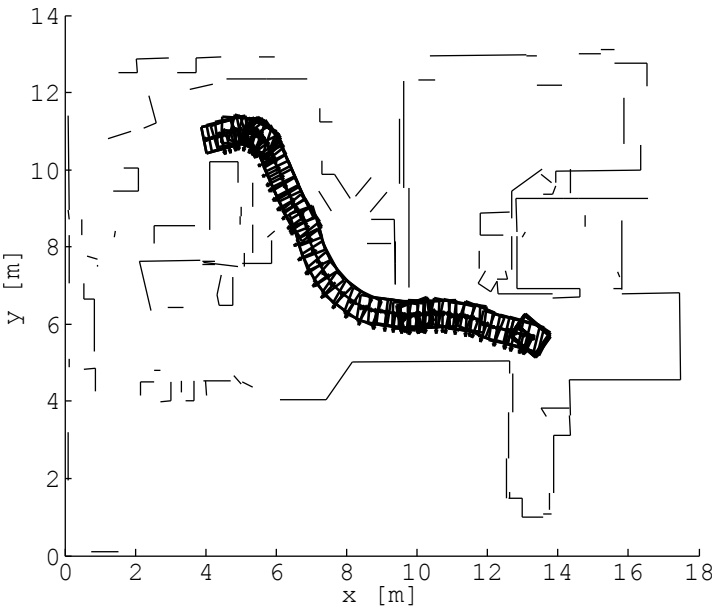


FIGURE 4.66: User's trajectory while travelling from the corridor back to the living room.

Significant overlap is present with previous driving. In particular, the current trajectory from corridor to living-room resembles the first manoeuvre displayed in figure 4.21. For much of the movement's duration, the user is driving at maximal linear velocity with low angular velocity. The user model in figure 4.68 concentrates then around ground-truth. Whenever turning is necessary, the user slows down and turns almost on-the-spot: in the user's input plot, figure 4.67, indentations appear on the linear input at the short-duration angular velocity peaks. The user model's likelihood in figure 4.68 shifts for these situations to high-curvature plans, which locally deviate from ground-truth in a similar way as in the first manoeuvre.

Also for this manoeuvre, the posterior (figure 4.69) reduces the user model's uncertainty. Previous user plan estimates are combined with current information in order to estimate the user's intention more accurately.

Despite all similarity of the current manoeuvre with previous driving, situation

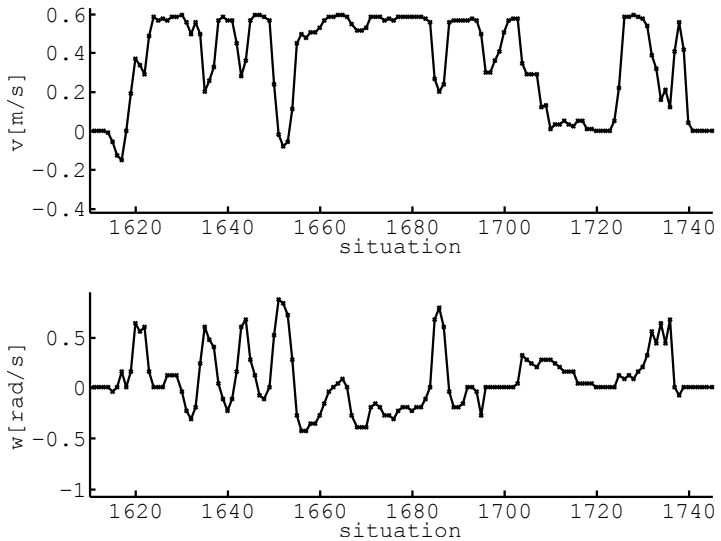


FIGURE 4.67: User's input while driving from the corridor to the living room.

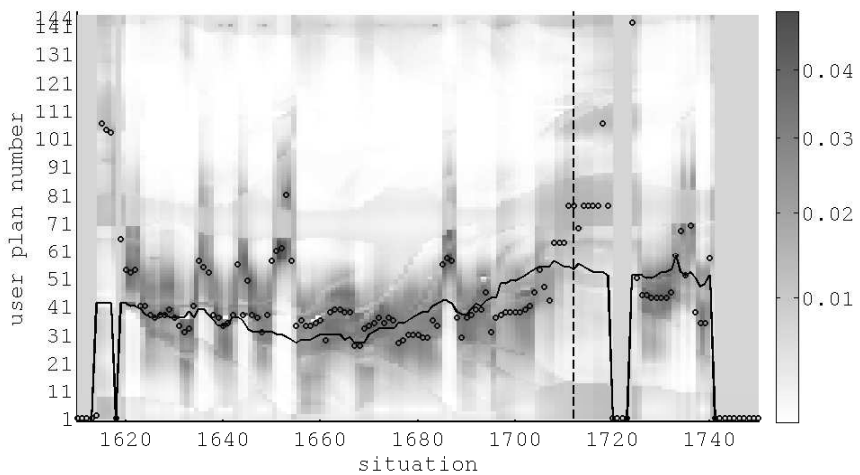


FIGURE 4.68: User model while driving from the corridor to the living room.

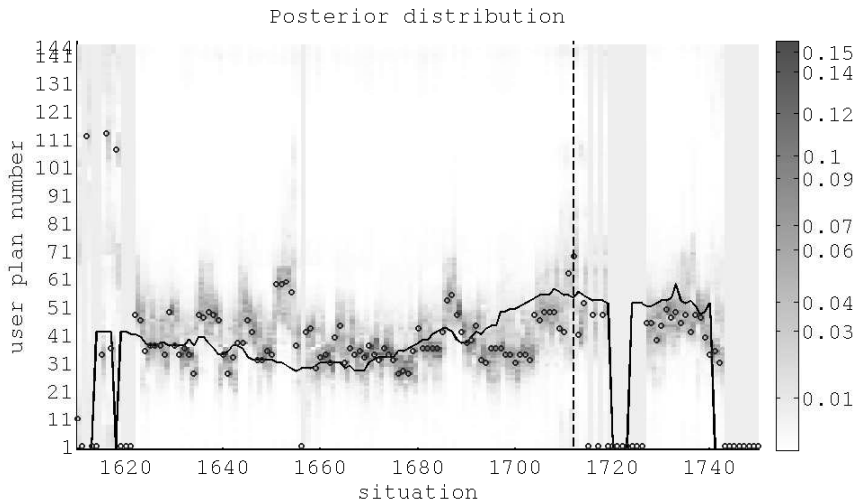
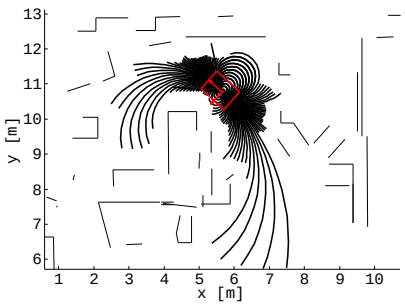
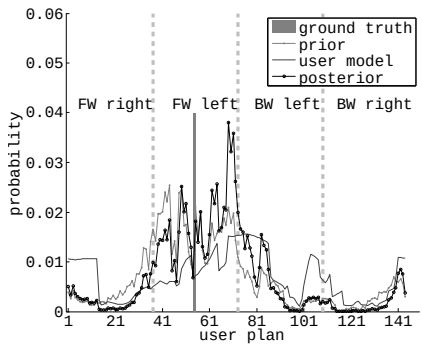


FIGURE 4.69: Posterior distribution over user plans while driving from the corridor to the living room.

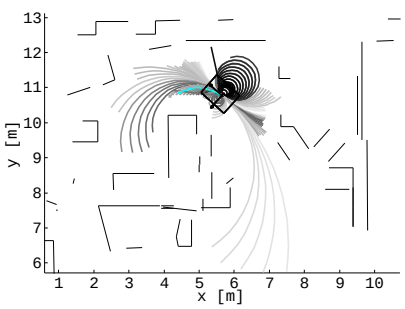
1712 deserves special mention. Parallels between time-step 303 of figure 4.33 and situation 1712 in figure 4.70 are evident. In both cases, the user's low-magnitude input results in highly uncertain user models. The probability graph for time step 1712 (figure 4.70(b)) shows that all plans are almost equally likely in the user model. Intermediate and low right curvatures forwards and backwards, as well as backwards intermediate left curvatures, are however, less likely. Unlikely plans cause holes in the mid-curvature region of the spatial user model plot, figure 4.70(c). As the user is pointing the joystick left, right-turns are less likely, unless their curvature is very high. For such curvatures the complete trajectory is collision-free, which increases their estimated mean angular velocity (reduced absolute value). Although still negative in value, these mean plan velocities enter the region within one standard deviation of the user's angular input. The posterior reduces part of the user model's uncertainty by assigning most probability mass to forward left-turning arcs. However, as the posterior plot of figure 4.70(d) reveals, many plan hypotheses remain probable. In the probability graph, figure 4.70(b), a broad prior and broad user model combine to a rather uncertain posterior. Previous user plan estimates and current information are too noisy to discern the user's true plan at time step 1712.



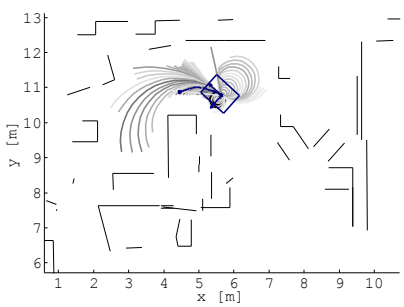
(a) all trajectories



(b) probabilities



(c) user model (future path in light blue)



(d) posterior

FIGURE 4.70: Low velocity input at time step 1712.

Discussion

Encouraging results for plan recognition and user modelling have been achieved in estimating the driving intention of a user diagnosed with spastic-quadriplegia. During the largest part of the experiments the robot could anticipate the user's plan and predict her future path.

In a detailed discussion we have shown how the Gaussian Process ($\mathcal{GP}$) user model captures the user's instantaneous driving style. Plan recognition enhances the inference of the user's true intent even further. For instance, this particular user prefers turning on-the-spot whenever changes in direction are necessary. The user model places then considerable probability mass around high-curvature trajectories, advancing forwards and backwards. Only through plan recognition, which considers prior driving, it is possible to infer the user's desired moving direction.

An additional goal of the extensive evaluation was to analyse the challenging situations for user modelling and plan recognition. Sometimes, the user model's uncertainty remained high. This is in part due to a high plan density, with many geometrical paths potentially explaining the user's noisy input signals. Therefore, reasoning about the temporal evolution of user plans becomes essential. At small input values the problem is especially acute. On the one hand, the standard deviations of the user model are large compared to the inputs' magnitude. On the other hand, plans become less distinguishable at low speeds because their density increases.

Another problem of the user model is its bias towards intermediate trajectory lengths for high curvature plans. Although there exist several situations where the user is turning on-the-spot, globally, high curvature turns are insufficiently represented in the training data. Calibration relies on ground-truth, which corresponds to the user's long-term driving destination. If the user turns rapidly, locally, a discrepancy between the user's long-term goal and her immediate driving intention is possible. We have highlighted examples of such situations throughout the experimental section. These discrepancies pose a problem when calibrating the user model since ground-truth differs from the user's instantaneous driving. As a consequence, the user model for high-curvatures is slightly biased towards intermediate trajectory lengths, which constitute the majority of the driving examples for turn on-the-spot data.

At low curvatures the bias is somewhat reversed and the user model favours longer paths, especially when the input of the user is large for the available free space. This can lead to a bi-modal user model when the shorter path lies straight-ahead. Please remember that the divide between left and right turns occurs at zero curvature. Arguably, this effect is desirable for navigation assistance. The robotic wheelchair, which employs the user model for decision making, would favour longer paths for high magnitude input. Still, the robotic wheelchair would need to decide between turning left or right. In this case, two shared control strategies are possible, depending on the estimated plan of the user. The robotic wheelchair could either

wait-and-see until more information becomes available. Alternatively, it could opt for an action, which disambiguates competing plan hypotheses, i.e. modes in the plan recognition distribution.

In the discussion of the experiments we have also addressed the level of noise contained in the plan process function, which defines the plan recognition process. An overconfident plan process function is undesirable in case of convergence to a wrong plan. Therefore, in order to increase robustness, we have opted to maintain a certain amount of noise during plan recognition.

Lastly, it is important to note the impact of a lack of backwards-driving data. Driving backwards was considered too dangerous for the test user and is therefore less represented in the training data. A partial remedy involved including a certain amount of artificial examples in the calibration of the user model. Without damaging forward-driving predictions, the virtual data allowed to distinguish forwards from backwards driving in the user model.

4.4 Conclusions

The main theme of this chapter is the wheelchair user, who we described from three viewpoints. The first perspective aimed at segmenting the potential user group of robotic wheelchairs. We provided a description of the most common pathologies and injuries of the nervous system and explained what impact these ailments have on navigation.

The second perspective presented the user from a system-theoretic viewpoint for manual control situations. As the literature review demonstrated, most prior work on manual control deals mainly with human models for compensatory control. However, wheelchair navigation must be understood beyond classic models for compensatory tasks. Wheelchair users also perform high-level cognitive process during navigation. They combine knowledge about the wheelchair with information about the environment and their intentions to create navigation plans according to an individualised strategy.

Therefore, we proposed a third perspective, the *probabilistic* view. The aim of this perspective is to endow the robotic wheelchair with reasoning capabilities about the cognitive processes of its user during navigation. We claimed that intuitive wheelchair control is possible only if the wheelchair understands the specific driving style of its user. In practice, such understanding means that the robot knows how a user transforms her plans into inputs to the robot. We formalised this notion in a probabilistic *user model*, which expresses the likelihood of the user's input provided this user has a certain plan in mind.

We discussed two user modelling approaches. In the first approach we derived an *analytical user model* for a user with weak directional steering. The interest of this model will become clear in the following chapter, where we show that thanks to

user-specific navigation assistance this user can reach intentions at the left when intended, even if the user is unable to turn in this direction. An interesting aspect of the analytical user model was its derivation. We divided the model for driving into an intrinsic stochastic component on the one hand, and the user-specific abilities on the other hand. This separation allows a comparative evaluation of the user's driving. The user's performance was related to an ideal reference user, who could execute any trajectory, albeit subject to noise.

Although, assessing driving performance comparatively might be important, obtaining such model in practice is difficult. Therefore, the second user modelling approach relied on a probabilistic machine learning technique called *Gaussian Process Regression* to derive user models directly from driving examples. We justified the selection of this technique by analysing user modelling requirements and by addressing how specific choices of the model fulfilled them.

We validated the *Gaussian Process* ($\mathcal{GP}$) user model in a case study of a user diagnosed with spastic-quadriplegia. During these experiments, the user had to maintain a very high level of concentration in order to fulfil a set of navigation tasks. Despite her best efforts, collisions with the environment still occurred. Therefore, this user could clearly benefit from a robotic wheelchair, which would assist in the most dangerous or tiring situations. Such a robot should ideally offer high driving comfort while, at the same time, be intuitive and easy to control. In order to meet these requirements, the robotic wheelchair needs to estimate the plan of the user by understanding her particular driving style. Encouraging results for plan recognition and user modelling were achieved in estimating the spastic user's driving intention. During the largest part of the experiments the robot could anticipate the user's plan and predict her future path.

Chapter 5

Shared wheelchair control

> *"When you're involved in a project like that, you work not only on the moment that is immediately present, but are constantly keeping in mind everything that has already happened, what must happen in the future, everything that can go wrong, and how that has to be coped with and what will in that case have to be prepared, if you are to avoid it slipping through your fingers. You can compare it with a war: in retrospect in the history books it's a nice, rounded story, the result of which is known; but while it was going on, you as a field marshal may have had your plan of campaign, but it was still largely a chaotic succession of events, stupidities, and unforeseen surprises, which demand new decisions every moment"*
>
> Harry Mulisch ($\star$1927 – †2010)
> *The discovery of Heaven*, page 196 (Mulisch, 1998)

Intuitive human-robot navigation is only possible if the robot respects the user's wishes. Since in most cases the user does not communicate her plans explicitly, the robot needs to estimate them. Estimated plans have to be translated then into robot actions. This implies a choice because merging different plans according to their estimated uncertainty might yield unwanted behaviour. This chapter proposes a shared control framework to formalise the notion of choice between safe robot actions that respect the wheelchair's kinematics and dynamics.

The following sections show how different human-centred desiderata, such as *collision-avoidance*, *obstacle-avoidance* or tracking the estimated plan of a user can be formalised as dynamic window approaches. In all cases, the basic building block for the different shared control algorithms are a set of local or global paths employed to predict user plans, as described in the framework for probabilistic human-robot navigation (section 3.2). Our shared control framework is sufficiently flexible to accommodate any type of user plan set. The only requirement is that

these plans are safe collision-free paths that respect the robot's kinematics. The dynamic window is then merely the means to consider the robot's dynamics.

We demonstrate the benefits of user-adapted shared control on a case study with a user who cannot steer to the left. Through navigation assistance, this user is able to reach locations at her left when she intends to do so.

5.1 Shared Control Algorithm

This section describes the main building blocks of all shared control algorithms we have implemented. First we explain how the wheelchair's dynamics are represented by the algorithms. Then follows an explanation of the basic dynamic window approach along with a relevant overview of prior dynamic window implementations in literature.

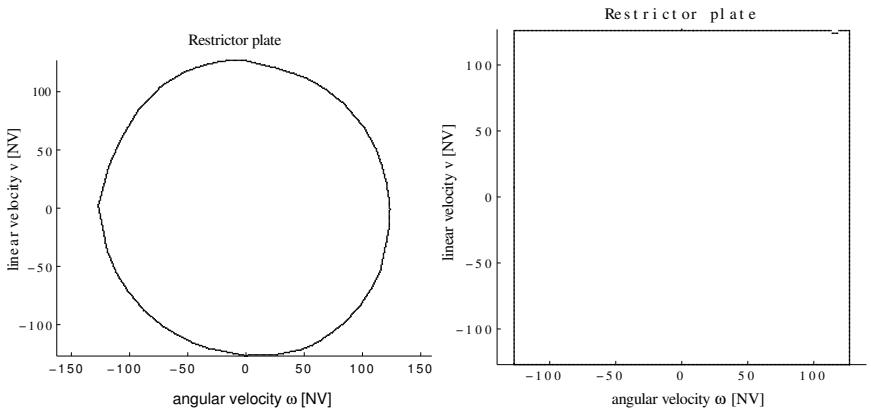
5.1.1 Representation of the wheelchair's dynamics

Restrictor plate The maximal deflection of standard wheelchair joysticks is physically restricted by the hardware in order to avoid simultaneously high linear and angular speeds. Figure 5.1 shows the restrictor plates of LAURA[1]'s joystick and of a PC-compatible joystick to control the MLR Simulator (Vanacker, 2008). These plots have been obtained by slowly moving the joystick on the border of the restrictor plates while recording the instantaneous deflection as described in (Demeester, 2007). LAURA's joystick is a conventional wheelchair input device manufactured by Curtis Instruments (Curtis, 2010). The MLRSIM joystick is a PC-gaming joystick lacking any restrictor plate. Given the physical constraints on the user's input imposed by the joystick, the shared control algorithm should be aware of the range of admissible speeds and only send these to the motors.

LAURA is a differentially driven mobile platform attached to a chair and controlled by a wheelchair joystick. LAURA's low-level control resembles more a mobile robot than a wheelchair since it lacks a configurable user-dependent smoothing of its control speeds. The implementation of a user-dependent smoothing for LAURA will be discussed in the next section.

In a first step of processing, the joystick deflection is read from the low-level communication interface every $80ms$ and converted to Network Variables [NV], ranging from -127 to 127. We adopt this convention for compatibility with the wheelchair Sharioto (figure 5.18) (Demeester, 2007; Vanhooydonck, 2007) equipped with a Dynamic Controls (Dynamic, 2010) control unit. The conversion from NV units to linear and angular speeds in SI units depends on the distance of the wheels which can change due to different pressure in the wheels, contact point to the floor,

[1]Leuven semi-AUtonomous Robotic Assistive wheelchair: L.AU.R.A



(a) LAURA's joystick restrictor plate (b) MLRSIM's joystick restrictor plate

FIGURE 5.1: Restrictor plates of LAURA and of the MLRSIM joystick. Both linear and angular velocities are in NV units.

friction, and slippage. Therefore, the conversion factor should be estimated online for increased precision. However, we have adopted a static conversion linking the maximal platform speed of $1.26m/s$ to the maximal range of $127NV$, and similarly linking the maximal rotational speed of $5.04rad/s$ achieved when turning on the spot counter-clockwise to $-127NV$. The sign reversal for angular speeds in NV again obeys the convention of Dynamic Controls.

Velocity profiles The joystick input of the user is not directly sent to the motors. Typically, a wheelchair controller smooths the input before applying it to the motors in order to avoid jerky driving. The controller from Dynamic Controls (Dynamic, 2010) smooths linear and angular speeds independently.

First, the joystick's deflection is scaled between a minimal and maximal value, which depend on the driving mode. A higher driving mode means faster speeds. Then, the scaled value is smoothed with a velocity profile. The wheelchair reaches the rescaled input twist of the user, $\mathbf{u}_{target}^{NV}$, gradually through a series of discretised twists $\{\mathbf{u}_{current}^{NV}, \mathbf{u}_2^{NV}, \ldots, \mathbf{u}_k^{NV}, \mathbf{u}_{target}^{NV}\}$ which are applied every $80ms$ to the motors. Smoothing of the user's input depends on the maximal linear and angular accelerations, the necessary amount of acceleration and a time constant. Figure 5.3(a) shows the evolution of motor signals for the linear speed as a response to a square joystick input wave, and figure 5.3(b) shows the same plot for the angular speed.

Physiotherapists adjust the parameters of the velocity profiles for each user according to her abilities. These parameters are expressed as percentages and are stored in the personalised user interface. In the example of figure 5.3 the user is

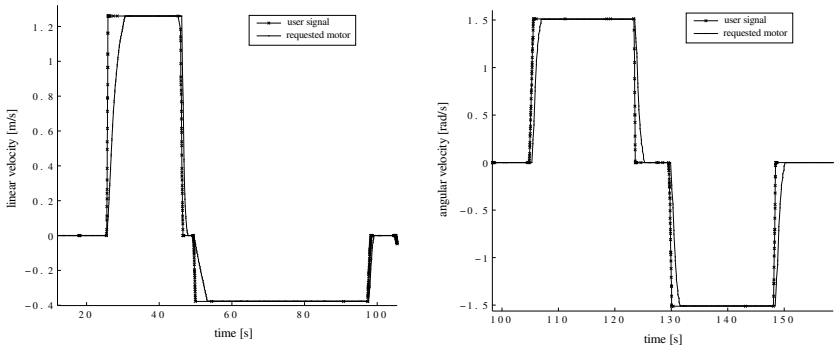


FIGURE 5.2: K.U.Leuven's wheelchair LAURA *Leuven semi-AUtonomous Robotic Assistive wheelchair*. LAURA is a differentially driven mobile platform attached to a chair and controlled by a wheelchair joystick. A SICK laser scanner allows LAURA to perceive her environment.

allowed to drive with maximal speed forward and with 30% of the maximal speed backwards. Turning to the left and to the right are also permitted at 30% of the maximum turning speed of $5.04\frac{rad}{s}$. Typically, as is also the case in the example, accelerations are softer than decelerations since the wheelchair should be able to stop as fast as possible.

5.1.2 Dynamic Window Approach

We have selected the dynamic window approach as our shared control algorithm because it explicitly considers the kinematics and dynamics of the robot. This



(a) Advancing forward and backwards. (b) Turning left and right.

FIGURE 5.3: Velocity profiles for the linear and angular speeds comparing the target joystick input to the motor speeds.

algorithm searches directly in the robot's velocity space for optimal controls. It simplifies the general trajectory planning problem, which grows exponentially with the number of considered time steps, to a search for the optimal velocity reachable within the next control step. Due to the limited acceleration capabilities of the robot, only certain velocities are reachable in a control interval. These reachable velocities form the *dynamic window* that gives its name to the algorithm. As the search for the optimal control is repeated for each time interval, the robot can achieve approximately any motion. Trajectory segments are approximated as a circular arcs and the complete robot motion corresponds then to a sequence of such arcs.

Figure 5.4 explains how the dynamic window is calculated for our robotic wheelchair. The left side of the figure shows the acceleration and deceleration velocity profiles described in the preceding section. The right side of the figure represents the dynamic window in velocity space, where the current twist is marked as a filled rectangle. In order to obtain the maximal extension of the dynamic window, the robot checks which velocity it could reach in the time interval ΔT if $\mathbf{u}_{target}$ was the maximal or minimal twists. Since our wheelchair can control independently linear and angular speeds, the dynamic window is rectangular but not necessarily symmetric concerning linear acceleration and deceleration. The grid representing the dynamic window indicates that only a discrete set of velocities are considered in the search for the optimal control. For our wheelchair the discretisation corresponds to the NV variables of the wheelchair controller from Dynamic Controls (Dynamic, 2010).

For computational efficiency, the extension of the dynamic window is pre-computed for every possible input velocity and stored as a look-up table.

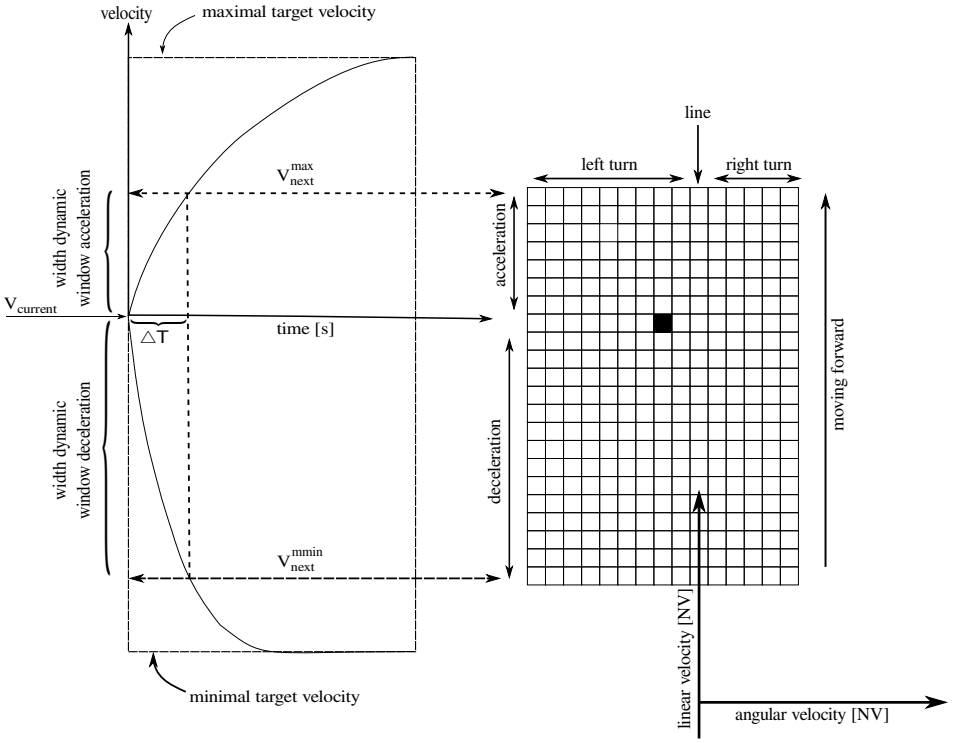


FIGURE 5.4: Dynamic Window

Admissible velocities

The robot chooses a twist that optimises a given cost function among a set of admissible twists within the dynamic window. A twist is admissible if the robot can stop in time without colliding along the trajectory resulting from executing the velocity. Figure 5.5 indicates how a safe stopping trajectory of a certain radius of curvature is calculated. Let us assume the current dynamic window is the rectangle indicated by number 1 on the right part of the figure and the stopping trajectory should be calculated for the velocity shaded in grey. This trajectory consists of a series of twists executed during the control time ΔT, $\mathbf{t}_{stop} = \{\mathbf{v}_1^{stop}, \ldots, \mathbf{v}_N^{stop}, \mathbf{0}\}$, bringing the robot to a stand-still.

Each twist of the stopping trajectory should have the same radius of curvature $\rho_{stop} = \frac{v_i}{\omega_i}$ as the twist under consideration, $(v, \omega)_i$. This property is exploited when calculating the sequence of stopping twists. The robot checks where the radius of curvature intersects the current dynamic window and adds the intersecting velocity that is closer to the velocity space's origin to the sequence of stopping twists. The newly added velocity represents the centre of the next dynamic window

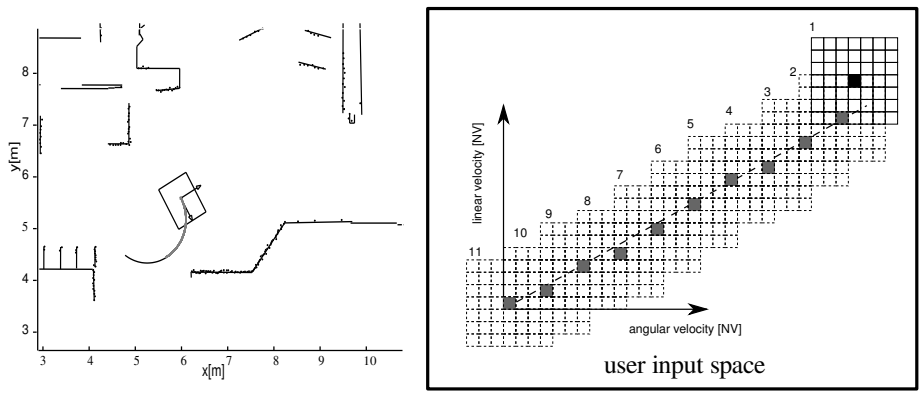


FIGURE 5.5: Calculation of a safe stopping trajectory with the same radius of curvature as a certain velocity of the dynamic window

in the stopping sequence. The process is repeated until convergence, i.e. until a dynamic window contains the velocity $\mathbf{0}$.

For increased computational efficiency, the dynamic window borders are parametrised and stored together with the maximal extension of the dynamic window for each possible input velocity. Calculating all stopping trajectories for 450 dynamic window twists takes around $30ms$ on a Dell Precision laptop running Linux with an Intel Core Duo Processor @$2.6GHz$ and $4GB$ of RAM.

On the left part of figure 5.5 it can be verified that the wheelchair is able to stop in time for the selected velocity. The collision-free trajectory in black contains the safe stopping trajectory.

Our implementation of the dynamic window approach selects an optimal control among admissible twists. This trades-off trajectory-following behaviour with executing the velocity desired by the user while avoiding at the same time collisions at high speed. The shared control desiderata are formalised as a linear cost function the robot minimises:

$$\begin{aligned} cost(v_i, \omega_i) = \alpha_{user} \cdot c_{user}(v_i, \omega_i) + \alpha_{path} \cdot c_{path}(v_i, \omega_i) \\ + \alpha_{coll} \cdot c_{coll}(v_i, \omega_i), (v_i, \omega_i) \in \mathbf{dwa}(k) \cap \mathbf{adm}(k) \end{aligned} \tag{5.1}$$

where c_{path} represents either collision-avoidance, obstacle-avoidance behaviour, or the cost of following a path estimated as the recognised plan of the user. In equation 5.1, c_{user} favours twists closer to the user's input and the term c_{coll} increases for collisions at high speed. $\mathbf{dwa}(k)$ is the dynamic window at time k and $\mathbf{adm}(k)$ the admissible velocity set.

User input

The user cost in equation 5.1 measures how much a certain dynamic window twist differs from the user's input. Since the raw joystick deflection corresponds to target velocities, the user's input is first converted to immediately executable velocities by means of the velocity profiles (cf. section 5.1.1).

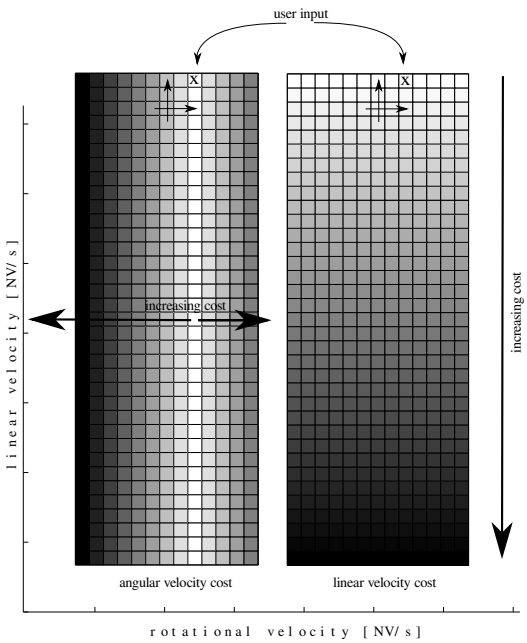


FIGURE 5.6: Cost related to the user's input, on the left for the angular speed and on the right for the linear speed.

Figure 5.6 shows the user related costs for the angular and linear speeds. The cost represents the L_1 distance for each velocity coordinate. The user's angular speed cost varies vertically and the linear cost varies horizontally. In figure 5.6 perpendicular arrows indicate the current twist and a cross marks the user's input.

It is possible to assign different weighting to each component in order to increase the importance of one velocity coordinate with respect to the other. However, we have opted for equal contribution of both costs.

Collision speed

A collision cost arises if the robot is unable to stop in time after processing motion commands at the bandwidth of the user. This cost is proportional to the speed at collision and corresponds to the term c_{coll} in equation 5.1.

Figure 5.7 illustrates the process of calculating the collision cost. The upper part of the figure shows the current dynamic window centred at the twist $(\omega_{NV}, v_{VN}) = (31NV, 96NV)$. The asymmetry in acceleration and deceleration in this particular case stems from the fact that the robot is approaching its configured maximally permissible rotational velocity of $38NV$ or $1.508\frac{rad}{s}$.

For this example we have chosen low deceleration capabilities in order to accentuate the effect of the collision speed cost. The solid lines on top of the dynamic window are trajectories with constant radius of curvature, and the dashed lines correspond to decision boundaries for collision checking in velocity space. All dynamic window twists within two dashed boundaries are assigned to the solid trajectory contained inside the dashed boundary.

The lower part of figure 5.7 displays the robot in the environment. We number all trajectories with precise collision information, which match the solid lines in the dynamic window of figure 5.7. An arrow points to the end of collision-free trajectory 5. The remaining trajectories, marked with special symbol and letters, illustrate the calculation of the collision speed cost for several dynamic window velocities.

The robot calculates collision-cost trajectories for every dynamically feasible twist within the dynamic window. These trajectories consist of two parts, the user reaction part and a stopping segment. During the first part of the trajectory, the dynamic window twist is integrated for the user's reaction time according to the motion model of equation 5.2, where θ_0 is the initial orientation.

$$\Delta x = \begin{cases} \frac{v}{\omega}\Big(\sin(\omega\Delta t + \theta_0) - \sin(\theta_0)\Big) & v \neq 0, \omega \neq 0 \\ v\Delta t\cos(\theta_0) & \omega = 0 \end{cases} \tag{5.2a}$$

$$\Delta y = \begin{cases} \frac{v}{\omega}\Big(\cos(\theta_0) - \cos(\omega\Delta t + \theta_0)\Big) & v \neq 0, \omega \neq 0 \\ v\Delta t\sin(\theta_0) & \omega = 0 \end{cases} \tag{5.2b}$$

$$\Delta\theta = \begin{cases} \omega\Delta t & \omega \neq 0 \\ 0 & \omega = 0 \end{cases} \tag{5.2c}$$

The first trajectory segment represents how fast the wheelchair user can react to unforeseen events, assuming she needs a certain amount of time to sense the event, process it, decide about an appropriate response, and execute this response. According to the overview study by (Green, 2000) (please refer also to section 4.2.4) the reaction time for braking cars as a response to unexpected events ranges from $1.5s$ to $1.8s$. This reaction time includes $0.3s$ motion time for changing from throttle to brake pedal. Given the similarities of the tasks and considering that the

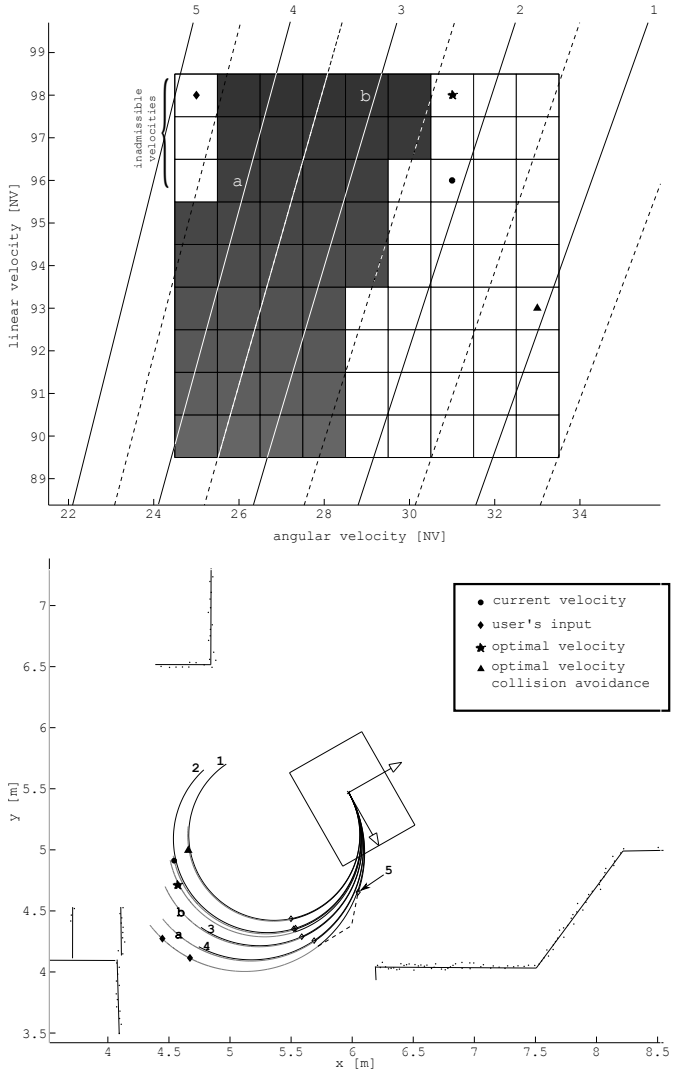


FIGURE 5.7: Cost for collisions at high speed. The top of the figure shows the current dynamic window. Trajectories of constant radius of curvature are displayed as numbered solid lines on top of the dynamic window. Dashed lines represent decision boundaries for collision-checking in velocity space. The trajectories employed to calculate the collision cost are marked with special symbols and letters in the environment and the dynamic window. These trajectories correspond to maintaining a certain dynamic window twist for the user's reaction time followed by a stopping trajectory segment.

braking reaction time measures the human response to an unexpected event and not the vehicle's dynamics or the vehicle's speed, we have opted to employ a similar value of $1.5s$ as estimate for the user's reaction time when driving a wheelchair.

For the second segment of the collision-cost trajectory the robot calculates a stopping trajectory as explained in section 5.1.2. Accurate collision checking information is only available for a discrete number of trajectories, indicated as solid lines in figure 5.7. Therefore, each dynamic window twist is assigned for collision-checking to the closest curvature radius, where proximity is measured in a polar representation of the curvatures.

The robot can stop in time if the stopping trajectory's length does not exceed the total length of the assigned collision-free trajectory. In order to check this condition, the robot computes the closest point on the assigned collision-free trajectory to the end-point of the first segment, which represents the trajectory travelled during the user's reaction time. Let this point be called $\boldsymbol{x}_i^{user} = (x_i, y_i, \theta_i)_{user}$, with i identifying the dynamic window twist (v_i, ω_i) under consideration. The closest point on the collision-free arc minimises the distance to $\boldsymbol{x}_i^{user}$ in equation 5.3.

$$d(\mathbf{x}_i^{user}, arc(\Delta t)) = \left[x_i^{user} - (x_0 + \Delta x_{arc})\right]^2 + \left[y_i^{user} - (y_0 + \Delta y_{arc})\right]^2 \quad (5.3)$$

If the collision-free arc is parametrised according to the motion model in equation 5.2 by $(v, \omega, \Delta t)$, then the closest point on the collision-free arc to $\boldsymbol{x}_i^{user}$ can be calculated analytically as stated in equation 5.4.

$$\Delta t^\star = \begin{cases} \frac{1}{\omega} \arctan\left[\frac{(x_i^{user} - x_0)\cos\theta_0 + (y_i^{user} - y_0)\sin\theta_0}{\frac{v}{\omega} + (x_i^{user} - x_0)\sin\theta_0 - (y_i^{user} - y_0)\cos\theta_0}\right] \pm \frac{\pi}{\omega} & v \neq 0, \omega \neq 0 \\ \frac{1}{v}\left[(x_i^{user} - x_0)\cdot\cos\theta_0 + (y_i^{user} - y_0)\cdot\sin\theta_0\right] & v \neq 0, \omega = 0 \\ \frac{\Delta\theta_i^{user}}{\omega} & v = 0, \omega \neq 0 \end{cases} \quad (5.4)$$

where $(v, \omega, \Delta t^\star)$ are the velocity-parametrisation coordinates of the closest point on the assigned collision-free arc to $\boldsymbol{x}_i^{user}$.

The length of the remaining collision-free trajectory starting at point $(v, \omega, \Delta t^\star)$ is then compared to the stopping trajectory's length. In figure 5.7 the trajectories marked as a and b exceed their assigned collision-free trajectories 4 and 3 respectively. All dynamic window twists within the decision-boundaries of trajectories 3 and 4 are assigned collision costs proportional to their values. Trajectory 5 is inadmissible, since the wheelchair would be incapable of stopping even without considering the user's reaction time.

5.1.3 Related work

Original dynamic window approach

The dynamic window approach was originally proposed by (Fox et al., 1997). In its original formulation it is a purely reactive collision-avoidance method. It relies mainly on immediate sensor information to derive motion commands, as oposed to global methods which employ a complete model of the robot's environment for navigation.

In the case of the synchro-drive robot described in the paper, with indenpendently controllable linear and angular speeds, the dynamic window is rectangular and centred around the current velocity of the robot. The algorithm only considers safe velocities with respect to obstacles in the immediate proximity. Among the admissible velocities, the optimal target velocity maximises the robot's objective function. The objective function chosen in (Fox et al., 1997) includes a measure of progress towards a goal location, the forward robot velocity, and the distance to the next obstacle on a circular trajectory. This function trades-off moving fast towards a goal with the behaviour of moving around obstacles.

$$G(v,\omega) = \sigma\left(\alpha \times heading(v,\omega) + \beta \times dist(v,\omega) + \gamma \times vel(v,\omega)\right) \tag{5.5}$$

$Heading$ measures the alignment of the robot with a goal, $dist$ is the clearance to the closest obstacle intersecting with a curvature, and vel defines the forward velocity of the robot. These factors are smoothed via a function $\sigma : \mathbb{R} \mapsto \mathbb{R}$ in order to achieve more side-clearance from obstacles. The robot chooses a trajectory for which it can maximise its translational velocity and distance to obstacles while minimizing the angle to a goal relative to its current heading. The performance of the algorithm depends on the relative weighing of heading with respect to the distance and velocity factors. Low values of α allow the robot to move freely around obstacles. However, they might prevent the robot from reaching its goal behind narrow openings. On the other extreme, a high heading factor α brings the robot very close to objects before turning away which hinders smooth circumvention of these obstacles.

(Fox et al., 1997) tested their approach on a synchro-drive B21 Robot manufactured by Real World Interface Inc. in real world environments with speeds of up to $95cm/s$. The robot was equipped with 24 Polaroid ultrasonic distance sensors, 56 infra-red detectors, and a stereo camera. The robot models its local environment as an obstacle line field derived from the sensors relatively to its current position. This line field contains a perpendicular line to the main axis of each sensor beam at the distance measured by the sensor. The length of the line depends of the width of the beam at the measured distance. With this representation of the local environment, the robot was capable of navigating in an office. It successfully

performed manoeuvres such as turning sharply into a room after having advanced along a corridor, or avoiding obstacles while advancing straight-ahead.

The curvature-velocity method for local obstacle avoidance

The curvature-velocity method by (Simmons, 1996) was developed independently from the original dynamic window algorithm by (Fox et al., 1997), but both methods are very closely related. The curvature velocity method formulates local collision avoidance as a constrained optimization in velocity space. Constraints are derived both from physical limitations of the robot, namely its maximal acceleration and maximal velocities, as well as from obstacles in the environment. The robot selects a motion command which maximises an objective function trading off speed, safety and goal-directness, while satisfying physical and environmental constraints. The curvature velocity method allows the robot to navigate safely in the presence of noisy sensors and dead-reckoning error. Furthermore, it avoids obstacles while advancing to the goal and is computationally efficient. Additionally, the method explicitly considers the dynamics of the robot, maximises the forward progress and controls both the speed and direction of the robot. The latter three requirements are not always addressed in obstacle avoidance algorithms.

The method assumes that the robot can control translational and rotational velocities, but it is non-holonomic, i.e., it cannot turn instantaneously but rather moves along circular arcs. This formulation disregards the effects of acceleration on the movement of the robot. However, for most indoor mobile robots travelling at low speed and capable of high accelerations, such an approximation is perfectly valid.

In order to derive environmental constrains in velocity space, the robot calculates how far it can travel along a certain arc before hitting an obstacle that is perceived by the sensors. Since obtaining the collision distance for each curvature is computationally infeasible, the curvature velocity method approximates the distance calculation by subdividing the velocity space in regions of constant distance to impact. The method simplifies the distance calculation further by approximating obstacles as circles. For each circular obstacle, the robot obtains the minimal and maximal tangential curvatures, which define a curvature interval. The robot further subdivides these intervals into quadrants starting from the closest point in metric coordinates to the robot's origin. Each quadrant forms then a piecewise approximation to the real impact distance. Overlapping curvature regions of several obstacles are assigned the overall minimal distance. Adding maximal velocities and accelerations constraints leads to triangular regions of constant impact distance.

The objective function maximised by the robot subject to physical and environmental constraints is a linear combination of three desiderata as indicated

in equation 5.6.

$$f(v,\omega) = \alpha_1 speed(v) + \alpha_2 distance(v,\omega) + \alpha_3 heading(\omega) \tag{5.6a}$$

$$speed(v) = \frac{v}{v_{max}} \; ; \; heading(\omega) = 1 - \frac{|\theta_g - \omega \cdot \Delta T|}{\pi} \tag{5.6b}$$

The speed term favours higher speeds all else being equal, the distance terms represents a preference for travelling a longer distance along a curvature without hitting obstacles, and the heading term decreases when the robot is aligned with a goal heading after turning at angular speed ω for a time interval ΔT. Since the objective function is monotonically increasing in the linear velocity v, and the distance value is constant within each curvature interval, equation 5.6 can be maximised very efficiently. The robot chooses the maximal value of the objective function among the upper vertexes of the curvature regions and the point with zero heading error.

The curvature method deals with sensor uncertainty and modelling errors by growing the perceived obstacles with a small safety margin and by further limiting the maximal speed proportionally to obstacle distances. The robot is only allowed to travel at full speed if it is at least a robot radius away from every obstacle. As the robot employs a local obstacle avoidance method, it might get stuck in local minima of the objective function. Those minima are avoided by rotating the robot away when the selected velocity command is too low.

(Simmons, 1996) tested the method on a synchro-drive robot from RWI equipped with 24 sonars and a 30°-field of view laser range sensor from Nomadics. Sonar and laser information were combined into a $20 - cm$ resolution histogram grid. The robot also provided dead-reckoning information at $8Hz$, which is the frequency at which the algorithm executed. The author determined experimentally that the algorithm is very sensitive to variations in heading weight, α_3, and relatively insensitive to the relative values of the distance weight, α_2, and the speed weight, α_1. The method performed badly for α_3 close to zero (aimless wandering) or if the ratio $\frac{\alpha_3}{\alpha_1+\alpha_2}$ was greater than 15%. In the latter case, the robot often got trapped in local minima. During experiments, the author also identified the need to encourage the robot to turn in the goal direction whenever the angular error in the third term was larger than $\pm 60^o$. This is achieved by increasing the weight of the goal error proportionally to the angular distance from the goal heading. The weight α_3 was replaced by the term $\alpha_3(1 + \alpha_4(\frac{\theta_g}{\pi})^2)$. Experimentally it was found that α_4 should be greater than zero and less than three in order to avoid local minima.

A hybrid collision avoidance method for mobile robots

(Fox et al., 1998; Burgard et al., 1999) propose a hybrid approach to collision avoidance integrating the strictly sensor based reactive dynamic window algorithm of (Fox et al., 1997) with information of an environment map.

The robot estimates its position in the map via an extension of the standard Markov localization. The Markov localization algorithm calculates a probability distribution over all possible positions, which in this case are represented in a finely-grained and equally-spaced grid with resolutions of $10cm$ and $15cm$ for the metric coordinates and 1° or 2° for the orientation. The robots deals with the huge number of discrete cells of large indoor environments of up to $30 \times 30m^2$ by storing the conditional position densities as look-up tables with extremely fast access, and by selectively updating only the most likely locations.

Besides describing an efficient algorithm for high resolution Markov localization, the authors have extended the basic algorithm to densely populated and dynamic environments. The presence of moving people violates the Markov assumption of the standard Markov localization algorithm, which requires the robot's position to be the only state. In order to accommodate moving objects, the robot selectively eliminates measurements that reduce the certainty about its current location estimate. The robot only accepts measurements, which reduce the entropy of the posterior position distribution compared to the prior. However, such measurement filtering might discard sensor readings that help the robot to recover from large localisation errors. Those measurements might be in disagreement with the current erroneous position estimate. Therefore, in order to increase the robustness of the localisation, the robot always incorporates a random subset of discarded measurements.

If the robot knows its position with high certitude, the navigation problem reduces to the standard dynamic window algorithm extended with information of a map. In the present implementation the robot combines dynamically build maps with a priori floor plans, which indicate undetectable obstacles.

If the robot is uncertain about its current location, the algorithm is guaranteed to avoid collisions with high probability. The robot includes a virtual sensor modelling its position uncertainty. Let $d_\alpha(\mathbf{x})$ be the distance of the closest obstacle when the robot is at location $\mathbf{x}$ and the virtual sensor is facing an angle α relative to the robot's orientation. $d_\alpha(\mathbf{x})$ can be precomputed using a map of the environment. If z_α is a random variable modelling the output of an α-oriented virtual sensor, the probability of z_α is given by a Dirac distribution [2] and $p(z_\alpha = d|\mathbf{x}) = 1$ if $d = d_\alpha(\mathbf{x})$ and zero otherwise. Furthermore, the unconditional density of $f_{z_\alpha}(z_\alpha = d)$ can be calculated as $f_{z_\alpha}(z_\alpha = d) = \int_{\mathbf{x}} f_{z_\alpha = d|\mathbf{x}}(z_\alpha = d|\mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}$ and the probability

[2] The probability density of the Dirac distribution is $z \sim f_{z_\alpha|\mathbf{x}}(z_\alpha|\mathbf{x}) = \delta(z_\alpha - d_\alpha(\mathbf{x}))$ and the probability of $z = d$ given the location $\mathbf{x}$ can be calculated as $p(z_\alpha = d|\mathbf{x}) = \int_{z=d-\epsilon}^{z=d+\epsilon} f_{z_\alpha|\mathbf{x}}(z_\alpha|\mathbf{x}) dz$.

$p(z_\alpha > d) = \int_{z>d} f_{z_\alpha}(z_\alpha)dz$. The virtual measurement is the largest distance $d^\star_\alpha$, such that with probability 0.99 a larger distance than $d^\star_\alpha$ is measured, namely:

$$d^\star_\alpha = \sup\{d \mid p(z_\alpha > d) \geq 0.99\} \tag{5.7a}$$

$$\begin{aligned} p(z_\alpha > d) &= \int_{\mathbf{x}} \int_{z>d} f_{z_\alpha|\mathbf{x}}(z_\alpha \mid \mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \int_{z>d} \delta(z - d_\alpha(\mathbf{x})) f_{\mathbf{x}}(\mathbf{x}) dz d\mathbf{x} \\ &= \int_{\mathbf{x}} f_{\mathbf{x}}(\mathbf{x}) \int_d^\infty \delta(z - d_\alpha(\mathbf{x})) dz d\mathbf{x} = \int_{\mathbf{x}} f_{\mathbf{x}}(\mathbf{x}) \cdot U(d_\alpha(\mathbf{x}) - d) d\mathbf{x} \\ &= \int_{\mathbf{x} \in \{\mathbf{x} | d_\alpha(\mathbf{x}) > d\}} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{5.7b}$$

The function $U(\cdot)$ in equation 5.7b is the Heaviside step-function resulting from integrating Dirac's impulse function as illustrated in the next equation.

$$\begin{aligned} \int_d^\infty \delta(z - d_\alpha(\mathbf{x})) dz &= \int_{t=d-d_\alpha(\mathbf{x})}^\infty \delta(t) dt = U(t) \Big]_{d-d_\alpha(\mathbf{x})}^\infty \\ &= 1 - U(d - d_\alpha(\mathbf{x})) = U(d_\alpha(\mathbf{x}) - d) \end{aligned} \tag{5.8}$$

As derived in equation 5.7b, the probability of virtual sensor measurement z_α exceeding a certain value d is calculated by adding the probability mass of all location estimates with larger obstacle distance than d. Equation 5.7a defines then a conservative rule to obtain the actual virtual sensor value, which with high probability underestimates the true obstacle's distance. This way the robot can avoid obstacles even in the presence of location uncertainty. The efficiency to calculate virtual sensor values can be increased by considering only positions with probability above a threshold. The authors chose a threshold such that the subset of positions represent more than 0.99 of the position probabilities.

The robot described in (Burgard et al., 1999) also includes a global motion planner, which calculates minimum-cost paths to arbitrary goal locations in the environment. These paths are computed via modified value-iteration. Goal cells have an initial value of zero and all other cells are initialised to infinity. Then, non-target grid cells get updated iteratively with the minimal value of neighbouring cells:

$$V_{x,y} \leftarrow \min_{\substack{\Delta x=-1,0,1 \\ \Delta y=-1,0,1}} \{V_{x+\Delta x, y+\Delta y} + p(c_{x+\Delta x, y+\Delta y})\} \tag{5.9}$$

The update rule adds the value of a neighbouring cell to the cost for moving to it measured as the probability of the cell being occupied, $p(c_{x+\Delta x, y+\Delta y})$. Upon convergence, $V_{x,y}$ represents the cumulative cost of moving to the nearest goal.

Motion controls can be generated any time, even before value-iteration converged. The robot determines its direction of motion by steepest descent on the value function. Post processing ensures thereafter that the steepest descend path maintains a minimum clearance from walls. The robot updates its value function in parallel with choosing a motion direction.

The standard value iteration algorithm requires some additional extensions, since it is inapplicable to the path planning problem with dynamically changing maps. Standard value iteration is inefficient when the value function increases, since updating the neighbours of too small cell values can be arbitrarily slow (Burgard et al., 1999). Therefore, the value function must be reinitialised completely each time the map changes. Since the map is updated almost constantly, a complete value function reset is very inefficient. The robot implements two performance improvements over the complete value function reset. Firstly, only too small values $V_{x,y}$ are reset iteratively if their value is lower than a dynamic programming update, namely:

$$V_{x,y} \leftarrow \infty \text{ if } V_{x,y} < \min_{\substack{\Delta x=-1,0,1 \\ \Delta y=-1,0,1}} \{V_{x+\Delta x,y+\Delta y} + p(c_{x+\Delta x,y+\Delta y})\} \tag{5.10}$$

Secondly, a rectangular bounding box identifies those values which might change.

The two foremost properties of the extended value iteration planner, is that the algorithm is any-time, meaning that solutions of different quality are available at any time instant, and that the algorithm can deal with replanning from any location since a value function is computed for the complete environment. Any-place planning allows the robot to generate motion commands even if it finds itself in unexpected locations.

The algorithm was tested on a synchro-drive robot during a six day exhibit at the German Museum Bonn (*Deutsches Museum Bonn*) where the robot served as an interactive museum tour-guide. The robot was equipped with laser range finders, sonar, and infra-red sensors. Thanks to the navigation algorithm, the robot operated for approximately 47 hours and travelled a distance of $18.6km$. Out of the 2400 tour requests only 6 could not be fulfilled. The robot travelled, when possible, at $80cm/s$ and on average at a speed of $36.6cm/s$, mostly due to the presence of people which forced the robot to decelerate.

High-speed global holonomic dynamic window approach

The paper by (Brock and Khatib, 1999) describes a global holonomic dynamic window approach. Holonomic robots lack any restrictions on the direction of instantaneous acceleration. However, in order to keep the implementation computationally feasible, the robot searches for an optimal velocity within a restricted search space that still resembles holonomic kinematics.

The circular dynamic window of the robot described in (Brock and Khatib, 1999) consists of discretised velocities in polar coordinates in a global reference frame. Expressing velocities in a global reference frame decouples the x- and y- coordinate axes. The robot follows a quadratic curve until reaching its desired velocity, as indicated in equation 5.11.

$$x(t_i) = x(0) + v_x t_i + \int_0^{t_i} a_x t dt = x(0) + v_x t_i + \frac{1}{2} a_x t_i^2 \tag{5.11}$$

In the three dimensional search-space of triplets (v_x, v_y, a_{xy}), assuming $a_x = a_y$, the robot discards unsafe velocities which prevent it from stopping in time. If t_i is the duration of the motion following a certain trajectory $(v_x, v_y, a_{xy})_i$, then the trajectory's length can be computed as stated in equation 5.12.

$$length\left((v_x, v_y, a_{xy})_i\right) = \int_0^{t_i} \|\mathbf{v}_i\| dt = \int_0^{t_i} \sqrt{(v_x + a_x t)^2 + (v_y + a_y t)^2} dt \tag{5.12}$$

Only if the trajectory is sufficiently long for executing $(v_x, v_y, a_{xy})_i$ during one control interval and stopping afterwards, the motion command $(v_x, v_y, a_{xy})_i$ is considered safe. Among safe velocities, the robot chooses a motion command that maximises a linear combination of four terms. The first term is a local minimum-free potential function measuring the robot's advance to the goal. The second term favours high velocities far away from the goal and low velocities in the goal's proximity. The third term is a binary function indicating if the robot is in the goal region. The fourth term measures the expected decrease in the navigation potential after executing a motor command, thus favouring commands which quickly lead to the goal location. These terms are normalized to the $[0, 1]$ interval with linear weight factors.

In order to overcome the problem of local minima of (Fox et al., 1997) when calculating a trajectory to the goal, the robot includes connectivity information of the free space surrounding it. At the control loop frequency, the algorithm converts raw sensor data into an occupancy grid which models the free configuration space in the robot's proximity. The robot calculates a navigation potential with a wave-front propagation algorithm starting at the robot's goal and finishing at the robot's current location. Each cell in the occupancy grid is labelled with the L_1 distance to the goal. Since the navigation function is computed every time a motion command is selected, the robot can operate in unknown and unmapped environments.

The authors implemented the global holonomic dynamic window approach on the Nomad XR4000 omnidirectional robot from Nomadic Technologies Inc. This robot can move at $1.2\frac{m}{s}$ omnidirectionally with accelerations up to $1.5\frac{m}{s^2}$. The robot was

able of moving reliably in large office-like environments [3] with little manoeuvring space at velocities above $1\frac{m}{s}$.

Global dynamic window approach in 3D C-space for arbitrarily shaped holonomic and non-holonomic mobile robots

The work by (Demeester et al., 2005) is the most related to the dynamic window algorithm described in section 5.1.2. The authors propose an algorithm for online planning considering the robot's geometry, its kinematics, and dynamics explicitly. Thus, the algorithm is valid for different type of robots with different geometries and different kinematic and dynamic constraints. The algorithm achieves smooth and safe driving behaviour also when the manoeuvring space compared to the robot's size is small. Such situations can only be tackled if the geometry as well as the kinematics and dynamics are modelled accurately. Furthermore, the global dynamic window approach ensures the safety of the robot, since the algorithm is robust against sensor uncertainty and inaccurate motor control.

The algorithm proceeds in two stages. Initially, the robot calculates a navigation function from a set of goal poses to the robot's current pose, where pose denotes the metric coordinates x and y as well as the robot's orientation θ. The navigation function comprises edges between all $3D$ cells that respect the robot's kinematics and point in the direction of the minimum cost path to a goal. In the second stage, the robot selects a velocity that is dynamically feasible and follows the minimum cost path to the goal.

In the first stage, geometric path planning explicitly considers the robot's geometry and its kinematics. Starting from a $2D$-grid map, the robot determines the complete occupied configuration space (the occupied C-space). The robot first computes offline a local occupied C-space, which consists of all poses colliding with an occupied grid cell at the origin. This local occupied C-space is then shifted to all occupied positions in the $2D$-grid map. The robot performs an OR operation of the local occupied C-space with the grid cells surrounding the occupied grid cell in the map in order to obtain the complete occupied C-space. Online collision checking is very efficient and simply consists of checking if a certain grid cell in the global occupied C-space is blocked.

The algorithm attaches to each cell an intrinsic cost of being at that cell, which is related to the distance to the nearest occupied grid cell. A separate obstacle clearance grid is maintained for each discrete orientation. Each slice of the grid is calculated with a $2D$ wave front propagation algorithm with all intrinsic costs equal to zero, with the occupied cells as the goal set, and a transition cost equal to the Euclidean distance between cells. Additionally, for each discrete orientation, a look-up table of reachable successor poses is maintained. These successors

[3] The robot moved in unknown environments of sizes $10m \times 10m$ and $5m \times 5m$ respectively. Those environments were mapped with a SICK laser range finder in occupancy grids of resolution $5cm \times 5cm$.

correspond to the 4 or 8 neighbours in a $2D$-grid map. They are calculated by integrating a discrete number of kinematically feasible velocities over a maximum path length Δl_{max} or a maximum angle $\Delta\theta_{max}$. The successors are the discrete poses laying closest to the trajectories and they impose automatically the robot's kinematic constraints. As only the successors are employed for collision checking, the integration length or integration angle are small. Each successor also enforces a transition cost, c_T, which allows configuring different driving styles, such as a preference for driving forward or of turning along larger curvatures. The successor information is stored in a look-up table which is constructed offline. This table contains for each discrete orientation the relative position of the successors, the cost of the successors, and the input velocities leading to the successors.

Once the free C-space has been constructed, a wave front propagates from a set of discrete goal positions to the robot's current position via a breadth-first search algorithm. In order to speed up computations, this implementation of the global dynamic window employs multi-resolution planning. Furthest away from the robot, the wave starts in a low resolution $2D$ grid until it reaches the vicinity of the robot. The vicinity is further subdivided into a low resolution $3D$-grid containing a high resolution 3D grid in the immediate proximity of the robot. This approach is motivated both by computational efficiency as well as by the need of accurate control close to the robot in order to deal with narrow openings. Compatibility between the $2D$-grid and the low resolution $3D$-grid is ensured if the goal positions of the low resolution $3D$-wave correspond to the points where the $2D$-wave intersects the $3D$ grid. The orientation of the low resolution $3D$ goals is the gradient of the $2D$-grid at the $3D$ goal position. The high resolution $3D$ grid is also initialised with the search and obstacle information from the outer, low resolution $3D$-grid.

The second stage of the global dynamic window algorithm corresponds to a constrained optimisation in velocity space. Only velocities (v_i, ω_i) reachable within a control step, Δt, are considered in order to avoid the complexity of the general motion planning problem. These velocities conform the dynamic window. For a robot with independently controllable angular and translational speeds, the dynamic window is rectangular and centred around the robot's current velocity. Within the dynamic window, the robot discards velocities which would lead to a collision because the robot is unable to stop after executing such an inadmissible velocity for the control interval Δt. From the remaining admissible velocities the robot selects a control which maximises the alignment with the gradient of the navigation function, is close to a desired velocity, and avoids collisions at high speeds. The desired navigation performance is formalised as a minimisation of a linear combination of three cost terms.

$$c_{DW}(v_i, \omega_i) = \alpha_{grad} c_{grad}(v_i, \omega_i) + \alpha_{vel} c_{vel}(v_i, \omega_i) + \alpha_{coll} c_{coll}(v_i, \omega_i) \quad (5.13)$$

The gradient cost $c_{grad}(v_i, \omega_i)$ measures alignment with the navigation function. The velocity cost, $c_{vel}(v_i, \omega_i)$, can be shaped to favour different velocities depending

on the region, i.e. high velocities far away from the goal and lower velocities in the goal's proximity. On the other hand, the collision cost $c_{coll}(v_i, \omega_i)$ is used to moderate the robot's speed in cluttered environments. A velocity (v_i, ω_i) has high collision cost if a collision at high speed would occur due to inaccurate motor control or uncertainty in the robot's pose estimation. The collision cost is calculated by checking for collisions after executing the dynamic window velocity starting from several poses in the uncertainty ellipse around the current pose. Collision due to inaccurate motor control is considered for velocities in an uncertainty region around the current dynamic window velocity (v_i, ω_i). The collision cost is the highest value of all velocities in the uncertainty region.

In order to achieve smoother driving behaviour an additional extension is introduced. The robot checks if the radius of curvature needed to follow the navigation function is contained within the dynamic window after executing a velocity for two control intervals. In this way the risk to select too high velocities with bad gradient following behaviour is reduced.

The authors evaluated the algorithm in simulation and on the robotic wheelchair Sharioto equipped with a laser range scanner and ultrasound sensors. A two stage planning approach was adopted with a $2D$-grid of size $14m \times 18m$ and resolution $0.2m \times 0.2m$ and a $3D$-grid of size $5m \times 5m \times 22.5^o$ centred around the robot. The dynamic window contained around 300 velocities and was executed at $2.5Hz$. With this settings the method was successful in planning complex manoeuvres in environments with narrow openings and little manoeuvring space given the wheelchair's size. Sometimes it was observed that the shape of the path changed due to small location estimation errors. However, the algorithm was fairly robust to a wide range of parameter values that just have an impact on the shape of an otherwise collision-free path.

Discussion

The literature review on prior dynamic window implementations shows that this algorithm has a long tradition in robotics. It is a very elegant and efficient solution to trajectory planning, which respects the robot's kinematics and dynamics.

Prior algorithms can be divided into local or global approaches. Global techniques rely on complete models of the robot's environment to calculate a trajectory. Conversely, local approaches just need a reduced subset of the world to generate robot controls, at the expense of obtaining suboptimal solutions. Their advantage over global methods reside in their reactivity to unforeseen changes in the environment. However, the robot might become trapped in U-shaped environments, or local minima of the navigation function.

Prior implementations of the dynamic window focus mainly on fully autonomous robot motion. Their goal is that the robot reaches its destination as fast as possible while maintaining a certain clearance from obstacles. However, human-

robot navigation faces a completely different set of requirements because a user is transported by the robot. For instance, driving comfort and smooth navigation are imperative. Further, the user is part of the control loop and her wishes should be respected by the robot when avoiding an obstacle. The user might also sometimes wish to collide with obstacles at low speed in order to open a door or push a box. Therefore, we distinguish situations of *obstacle avoidance* from *collision avoidance*, which ensures the robot only collides at low speeds. In our framework, a safety layer of *collision avoidance* guarantees the safety of the user at all times.

Our implementation of the dynamic window algorithm considers user inputs and estimated user plans as part of the navigation function. This is imperative for intuitive human-robot navigation. Human-centric requirements such as the user's bandwidth or the user's reaction time are incorporated into the dynamic window as well. The shared control algorithm should not cut the user out of the control loop by taking decisions at a very high frequency. Additionally, the dynamic window should respect delays in the user's reaction time when calculating the optimal robot action. Any action should allow the user sufficient time to respond to unforeseen events, as well as guarantee that the robot can still stop in time.

As an application-specific requirement, our dynamic window is parametrised by the same set of parameters that physiotherapist normally adjust when adapting a wheelchair to the specific driving abilities of a user.

5.2 Collision- and obstacle-avoidance respecting kinematic and dynamic constraints

This section presents two standard algorithms for navigation assistance, *collision-avoidance* and *obstacle-avoidance*. Although these terms encompass a large variety of mobile robotic algorithms, not all are suited for human-robot navigation. For instance, the Vector Field Histogram method to obstacle avoidance (Borenstein and Koren, 1991) approximates the robot's dynamics and kinematics by assuming the robot can turn immediately in any direction. It also fails to consider the robot's shape. These approximations can lead to collisions with objects in the environment because the robot "cuts corners". Due to the wheelchair's large size it is important that navigation assistance algorithms model the kinematics and dynamics of the wheelchair accurately. Therefore, this section presents two implementations for obstacle and collision avoidance, which respect the robot's movement constraints while considering at the same time its shape.

Collision and obstacle-avoidance are basic shared control algorithms, which share the user's autonomy in order to improve her safety during navigation. They represent a first stage of shared control, since the user maintains considerable control over the robot. Thus, collision- and obstacle-avoidance are interesting for users who like to drive fast, or for users who need to navigate safely in cluttered environments but still want to maintain most of the control over the robot. These basic algorithms are of industrial interest because they offer basic navigation assistance without restricting the user's freedom considerably.

The main contribution of this section is a formulation of collision- and obstacle-avoidance in our framework for human-robot navigation. The implementation relies uniquely on a local set of safe user plans. Further, collision- and obstacle-avoidance with their restricted functionality serve as reference to compare more advanced shared control strategies. Collision-avoidance functions also as safety layer of more advanced shared control algorithms in order to avoid collisions at high speed.

5.2.1 Collision avoidance

In collision-avoidance mode the robot slows down before hitting an obstacle following the curvature prescribed by the user. We can formalise this behaviour as a specific cost function of a dynamic window. For the reader's convenience we restate the objective function minimised by the robot in the dynamic window approach.

$$cost(v_i, \omega_i) = \alpha_{user} \cdot c_{user}(v_i, \omega_i) + \alpha_{ca} \cdot c_{ca}(v_i, \omega_i) \tag{5.14}$$

The term for collisions at high speed, $c_{coll}(v_i, \omega_i)$, has been subsumed into the collision-avoidance cost since both terms impose similar behaviour on the robot.

In addition to avoiding collisions, the term c_{ca} matches the user's input as closely as possible. Still, the user's cost and the collision-avoidance term in equation 5.14 might contradict each other when the user is requesting a high speed which is unsafe in the surrounding environment. Therefore, by design, we impose that the collision-avoidance cost dominates always the user's cost in case a certain twist (v_i, ω_i) is unsafe. Such priority can be established by careful selection of the weighting terms α_{user} and α_{ca}.

Constraints on the weighting factors

Let $\mathbf{v}_{user}$ be the input of the user and $\mathbf{v}_{ca}$ the optimal collision-avoidance twist. The user cost of the user's twist is zero, $c_{user}(\mathbf{v}_{user}) = 0$. From this property we derive a constraint on the cost for user twist and optimal collision-avoidance twist in equation 5.15.

$$c_{ca}(\mathbf{v}_{user}) - c_{ca}(\mathbf{v}_{ca}) \overset{!}{>} \frac{\alpha_{user}}{\alpha_{ca}} \tag{5.15}$$

The derivation of the constraint in equation 5.15 is stated in the next set of equations, which consider $c_{user}(\mathbf{v}_{user}) = 0$.

$$\alpha_{ca} \cdot c_{ca}(\mathbf{v}_{user}) > \alpha_{user} \cdot c_{user}(\mathbf{v}_{ca}) + \alpha_{ca} \cdot c_{ca}(\mathbf{v}_{ca}) \tag{5.16a}$$

$$c_{ca}(\mathbf{v}_{user}) - c_{ca}(\mathbf{v}_{ca}) > \frac{\alpha_{user}}{\alpha_{ca}} \cdot c_{user}(\mathbf{v}_{ca}) \tag{5.16b}$$

$$\frac{\alpha_{user}}{\alpha_{ca}} \geq \frac{\alpha_{user}}{\alpha_{ca}} \cdot c_{user}(\mathbf{v}_{ca}) \tag{5.16c}$$

The first inequality, equation 5.16a, imposes that the cost of $\mathbf{v}_{user}$ exceeds the cost of the optimal collision-avoidance twist $\mathbf{v}_{ca}$. $c_{ca}(\mathbf{v}_{user}) - c_{ca}(\mathbf{v}_{ca})$ is always positive since $\mathbf{v}_{ca}$ minimises the collision-avoidance term. The latter inequality originates from a user cost normalised to the interval $[0, 1]$.

The simplest collision cost, which is compatible with the constraint of equation 5.15 while at the same time penalises collisions at high speed, is a set of linear functions stated in the next equation.

$$c_{ca}(\mathbf{v}_i) = \frac{1}{M_{ca}} \begin{cases} \max\left\{ M_v \cdot \frac{|v_i| - v_{th}}{v_{max} - v_{th}}, M_\omega \cdot \frac{|\omega_i| - \omega_{th}}{\omega_{max} - \omega_{th}} \right\} & \begin{Bmatrix} v_{th} < |v_i| < v_{max} \\ \omega_{th} < |\omega_i| < \omega_{max} \end{Bmatrix} \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

v_{th} and ω_{th} define the threshold linear and angular speeds at which collisions are still permissible. The user should be allowed to collide at low speeds, for instance, to be able to push doors open or to interact with objects in the environment such as cupboard doors. M_v denotes the maximal cost for the linear speed, M_ω the maximal cost for the angular speed, and M_{ca} the maximum of the collision-avoidance term. An example plot of the linear cost terms is provided in figure 5.8.

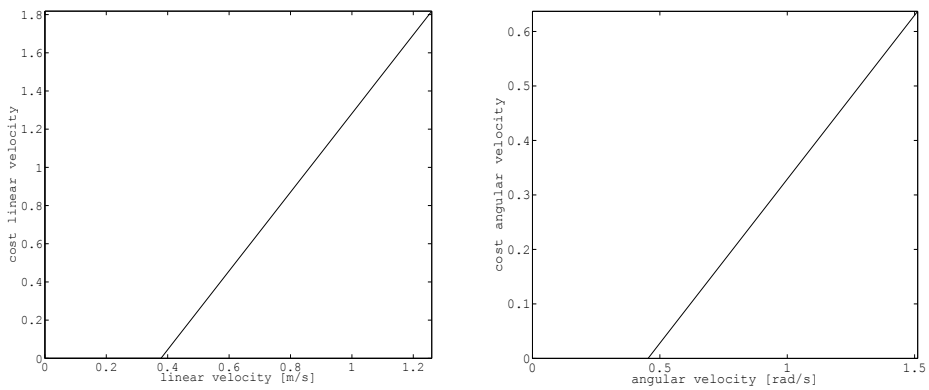


FIGURE 5.8: Linear costs collision avoidance as a function of linear and angular speeds.

Applying the constraint of equation 5.15 to the definition of the cost function for collision-avoidance in equation 5.17 yields two inequalities for M_v and M_ω.

$$M_v > \frac{\alpha_{user}}{\alpha_{ca}} \cdot M_{ca} \cdot \frac{v_{max} - v_{th}}{|v_{user}| - |v_{ca}|} \tag{5.18a}$$

$$M_\omega > \frac{\alpha_{user}}{\alpha_{ca}} \cdot M_{ca} \cdot \frac{\omega_{max} - \omega_{th}}{|\omega_{user}| - |\omega_{ca}|} \tag{5.18b}$$

These inequalities are unbounded when user and collision-avoidance twists coincide. However, for this case there is no need to establish a precedence relation. Precedence must hold when user twist $\mathbf{v}_{user}$ and collision-avoidance twist $\mathbf{v}_{ca}$ oppose each other maximally, i.e. when they are at opposite corners of the current dynamic window. Such precedence must be true for all possible dynamic window locations. Therefore, the smallest dynamic window width determines the relations between the parameters of the collision-avoidance cost function.

$$M_v \overset{!}{\geq} \frac{\alpha_{user}}{\alpha_{ca}} \cdot M_{ca} \cdot \frac{v_{max} - v_{th}}{\Delta v_{DW}^{min}} \tag{5.19a}$$

$$M_\omega \overset{!}{\geq} \frac{\alpha_{user}}{\alpha_{ca}} \cdot M_{ca} \cdot \frac{\omega_{max} - \omega_{th}}{\Delta \omega_{DW}^{min}} \tag{5.19b}$$

Equation 5.19 must further hold for $M_{ca} > \max\{M_v, M_\omega\}$ since the collision-avoidance cost is normalised to the interval $[0, 1]$. This property imposes an additional constraint linking the linear weights α_{ca} and α_{user} as follows:

$$\left.\begin{array}{l} \frac{\alpha_{user}}{\alpha_{ca}} \cdot \frac{v_{max}-v_{th}}{\Delta v_{DW}^{min}} < 1 \\ \frac{\alpha_{user}}{\alpha_{ca}} \cdot \frac{\omega_{max}-\omega_{th}}{\Delta \omega_{DW}^{min}} < 1 \end{array}\right\} \Rightarrow \alpha_{ca} > \alpha_{user} \cdot \max\left\{ \frac{v_{max} - v_{th}}{\Delta v_{DW}^{min}}, \frac{\omega_{max} - \omega_{th}}{\Delta \omega_{DW}^{min}} \right\} \tag{5.20a}$$

Calculation of the collision-avoidance cost

Figure 5.9 shows how the collision-avoidance cost is calculated. The top part of the figure displays the current dynamic window. The solid lines starting at $\boldsymbol{v} = (0, 0)$ represent dynamically feasible collision-free trajectories of constant radius of curvature in velocity space. Dashed lines denote decision boundaries for collision checking in velocity space. Similarly to the collision cost at high speeds described in section 5.1.2, accurate collision information is only available for a limited number of trajectories. Therefore, all twists of the dynamic window $\boldsymbol{v}_i$ are assigned for collision-checking to the closest curvature in a polar representation of the trajectories.

The twists in the top right corner of the current dynamic window are inadmissible because the robot would be unable to stop in time. The bottom part of figure 5.9 displays the robot in the environment together with the same collision-free trajectories of the top sub-figure. A set of special twists and their assigned trajectories are highlighted, namely the current twist, the user's input, the optimal twist according to the collision avoidance cost, and the overall optimal twist.

Collision-free trajectories are parametrised as triplets $(v, \omega, \Delta t)$ according to the motion model of equation 5.2, where Δt denotes the time to traverse the circular arc when the robot executes the twist (v, ω). In the current implementation, we opted for simple circular arcs as basic user plan hypotheses. However, the description of the collision-avoidance algorithm which follows is equally applicable to user plans composed of a sequence of such circular arcs with only minor extensions.

Let dynamic window twist (v_i, ω_i) be assigned to collision-free trajectory $(v, \omega, \Delta t)$. It is necessary to find a similar twist $(\omega_i, v_i)_{traj}$ which resembles the original twist, but is constrained to the same curvature radius of the assigned collision free-trajectory. We will denote $(\omega_i, v_i)_{traj}$ the collision-free equivalent to dynamic window twist (v_i, ω_i).

In collision-avoidance mode, this twist is calculated by maintaining the same angular speed as the dynamic window twist (v_i, ω_i), while keeping the trajectory's turning

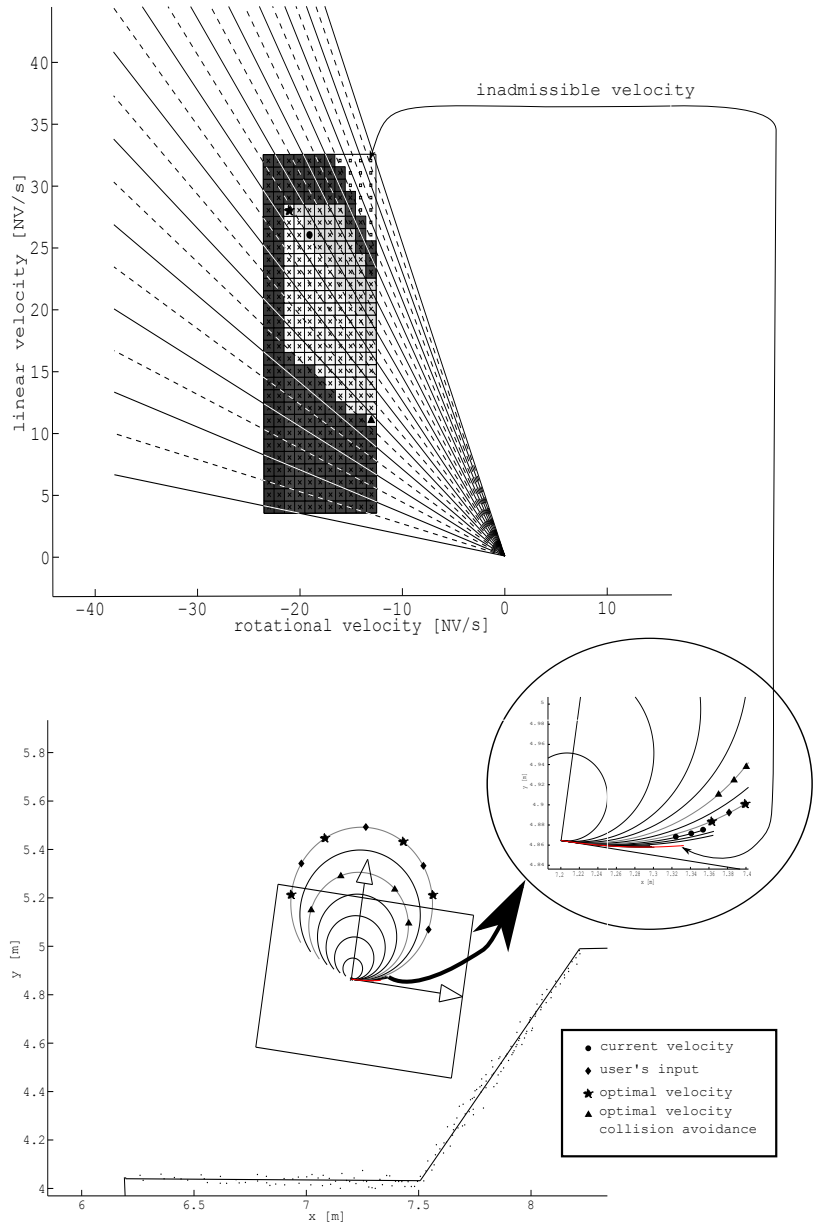


FIGURE 5.9: Dynamic window collision-avoidance displayed for a particular situation of the robot.

direction. The linear speed v_i^{traj} results from the prescribed radius of curvature

$\rho = \frac{v}{\omega}$ in case $\omega \neq 0$. Should the collision-free trajectory be a straight line instead of an arc, then the trajectory's angular speed equals zero while the advancing speed is preserved. Equation 5.21 summarises the conversion of a dynamic window twist (v_i, ω_i) to a compatible representation for collision-free trajectory $(v, \omega, \Delta t)$.

$$\Big(\omega_i, v_i\Big)_{traj} = \begin{cases} \Big(\text{sign}(\omega) \cdot |\omega_i|, \frac{v}{\omega} \cdot \text{sign}(\omega) \cdot |\omega_i|\Big) & \omega \neq 0 \\ \Big(0, \text{sign}(v) \cdot |v_i|\Big) & \omega = 0 \end{cases} \tag{5.21}$$

The sign$(\cdot)$ operator preserves the turning and advancing directions of the assigned collision-free trajectory.

Human factors in collision-avoidance The collision-free equivalent to twist (v_i, ω_i) in equation 5.21 allows to determine the travelling time of the robot along the assigned trajectory. This travelling time Δt_i^{travel} for twist $(\omega_i, v_i)_{traj}$ follows in equation 5.22.

$$\Delta t_i^{travel} = \begin{cases} \frac{\omega \cdot \Delta t}{\omega_i^{traj}} & \omega \neq 0 \\ \frac{v \cdot \Delta t}{v_i^{traj}} & \omega = 0 \end{cases} \tag{5.22}$$

On basis of the travelling time, the robot can decide whether the user would have sufficient time to react to an obstacle. If the user's reaction time exceeds the available travelling time, the trajectory is unsafe for the user and the robot should slow down. This is accomplished by penalising twists whose equivalent travelling time is shorter than the user's reaction time. According to the findings about human factors for braking cars, a reasonable estimate for the user's reaction time is $1.5s$ (Green, 2000) (please refer also to section 4.2.4). If the travelling time along a collision free trajectory is too short for a particular twist, then this twist has the collision-avoidance cost stated in equation 5.17.

The user's reaction time defines a hard threshold for the collision cost. In order to soften the condition on the travelling time, a gradual transition to zero collision-cost for travelling times ranging from the user's reaction time RT to $1.2 \cdot RT$ is introduced next.

$$\psi_{ca}(\Delta t_{travel}) = \frac{1}{2}\left(1 + \cos\left[\frac{\pi}{0.2 \cdot RT}(\Delta t_{travel} - RT)\right]\right) \tag{5.23}$$

The overall shape of the transition factor as a function of travelling time is shown in figure 5.10.

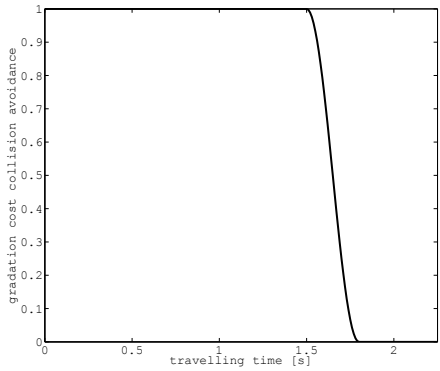


FIGURE 5.10: Decrease of the collision avoidance cost as a function of the travelling time.

Collision-avoidance cost The robot will execute the twist matching most closely the user's input while ensuring safe driving at the same time. Safe driving is measured in terms of available travelling time on a collision-free trajectory, as explained previously. The preference for following the user's input leads the robot to assign maximal cost M_{ca} to all twists exceeding the user's twist. Further, only twists in a neighbourhood of the user's input are evaluated according to the cost function of equation 5.17. The cost for twists outside the user's neighbourhood are rescaled linearly to the interval $[\max\{M_v, M_\omega\}, M_{ca}]$. Rescaling instead of assigning the maximal cost M_{ca} can prove invaluable if all twists within the user's neighbourhood are inadmissible. Then, the collision-avoidance cost still grades twists according to their safety. Equation 5.24 summarises the complete set of conditions on the collision-avoidance cost.

$$c'_{ca}(\mathbf{v}_i) = \begin{cases} c_{ca}(\mathbf{v}_i) & \mathbf{v}_i \in N(\mathbf{v}_{user}), \Delta t_i < RT \\ c_{ca}(\mathbf{v}_i) \cdot \psi_{ca}(\Delta t_i) & \mathbf{v}_i \in N(\mathbf{v}_{user}), RT < \Delta t_i < 1.2 \cdot RT \\ 0 & \mathbf{v}_i \in N(\mathbf{v}_{user}), 1.2 \cdot RT < \Delta t_i \\ M_{ca} & |v_i| > |v_{user}| \vee |\omega_i| > |\omega_{user}| \\ \frac{M_{ca}-M}{M} \cdot c_{ca}(\mathbf{v}_i) + M & \mathbf{v}_i \notin N(\mathbf{v}_{user}), \begin{matrix} |v_i| < |v_{user}| \\ |\omega_i| < |\omega_{user}| \end{matrix} \end{cases} \tag{5.24a}$$

where $M = \max\{M_v, M_\omega\}$, Δt_i is the collision-free travelling time for twist (v_i, ω_i), $c_{ca}(\mathbf{v}_i)$ is the cost function in equation 5.17, $\psi_{ca}(\Delta t_i)$ is the gradual transition defined in equation 5.23, RT is the user's reaction time, and $N(\mathbf{v}_{user})$ is the neighbourhood of the user's twist. We define a neighbourhood in velocity space as a constant number of collision-free trajectories surrounding a particular twist.

Figure 5.11 compares the different cost terms of the dynamic window for the robot in the situation of figure 5.9. We show the total cost function of the current dynamic window overlaid with the collision-free trajectories and the decision boundaries for collision-checking. The remaining plots represent the collision-avoidance cost and the two costs related to the user's input.

Clearly, the overall cost is mostly determined by the collision-avoidance term, as expected, given the choice of scale parameters α_{user} and α_{ca}. The influence of the user cost is strongest for those twists with zero collision-avoidance cost.

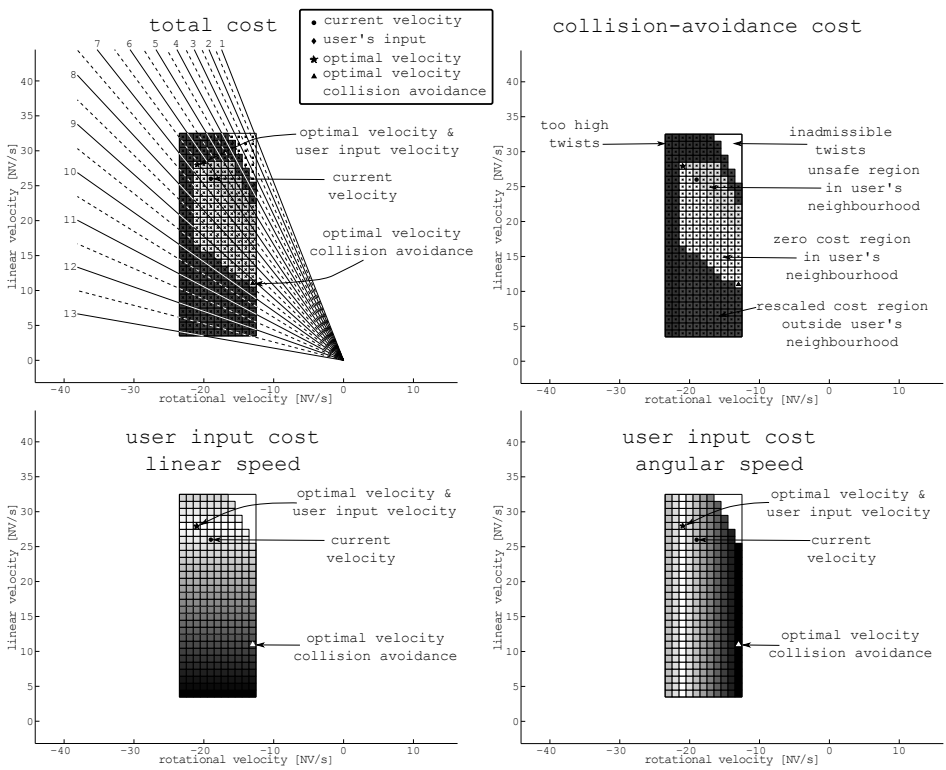


FIGURE 5.11: Cost collision-avoidance versus the cost for the user's input.

All areas of the collision-avoidance cost are visible in figure 5.11. The lighter regions represent the user's neighbourhood divided into safe and unsafe sectors. The rescaled region is indicated in dark grey and the twists exceeding the user's input are coloured black. Within the white region, all velocities have a zero cost for collision-avoidance since the assigned collision-free trajectories are long enough. In this region the user cost term defines which twist is the overall optimum, which in the example coincides with the user's input.

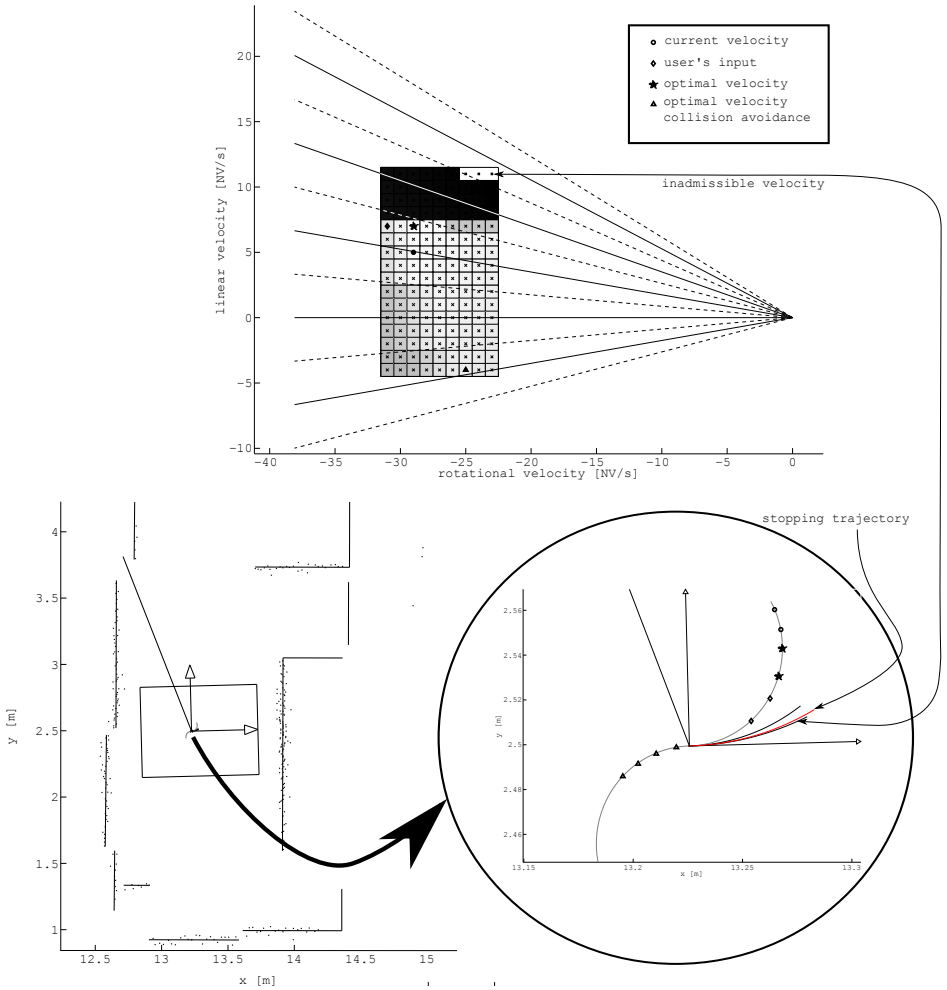


FIGURE 5.12: Total cost collision avoidance displayed next to the situation of the robot in the environment.

Figure 5.12 shows an additional snap-shot of collision-avoidance in action. The user is trying to turn at high speed in a very limited space. The same objects are represented as in figure 5.9, namely the total cost function for the current dynamic window along a set of collision-free trajectories available to the robot. In the plot of the environment a long straight line marks the final orientation for a turn on the spot (zero linear speed: $v = 0, \omega \neq 0$). The right part of the environment plot zooms into the short dynamically feasible trajectories. The colour gradation in the total cost varies vertically, with lighter colours for lower angular speeds. Again, the overall optimal twist differs from the optimal twist with respect to

the collision-avoidance cost. In regions of zero collision-avoidance cost, the term measuring proximity to the user's input dominates.

5.2.2 Obstacle avoidance

Obstacle-avoidance mode is a form of shared control where the robot circumvents obstacles instead of merely stopping in front of them as in pure collision-avoidance mode (please refer to section 5.2.1). However, in order to ensure the driver's safety, it is desirable to include also collision-avoidance in case circumventing obstacles becomes impossible for a certain situation. Obstacle-avoidance can be expressed within the dynamic window framework described in section 5.1.2. In this framework the robot is minimising the following objective function:

$$cost(\mathbf{v}_i) = \alpha_{user} \cdot c_{user}(\mathbf{v}_i) + \alpha_{oa} \cdot c_{oa}(\mathbf{v}_i) + \alpha_{coll} \cdot c_{coll}(\mathbf{v}_i) \tag{5.25}$$

The robot calculates the cost of every admissible dynamically-feasible twist, $\mathbf{v}_i \in \mathbf{dwa}(k) \cap \mathbf{adm}(k)$. $\mathbf{dwa}(k)$ represents the dynamic window which includes all velocities reachable within one control interval. Admissible velocities allow the robot to stop in time and their set is named $\mathbf{adm}(k)$. The term c_{user} favours twists in the proximity of the user's input, c_{oa} evaluates obstacle-avoidance behaviour, and c_{coll} penalises collisions at high speed. The choice of the mixing weights in equation 5.25 will be explained after the description of the obstacle-avoidance term, c_{oa}.

From an obstacle-avoidance perspective, longer collision-free trajectories are favourable. Thus, the obstacle-avoidance cost must increase for shorter trajectories. Further, not all trajectory lengths are equally important. Obstacle-avoidance becomes more pressing for shorter trajectories, whereas longer trajectories are almost equally desirable. Similarly to the collision-avoidance mode described in section 5.2.1, accurate collision-checking information is only available for a limited number of trajectories. They form the set of plausible user plans. In turn, each dynamically-feasible twist in the dynamic window is assigned to a collision-free user plan. The obstacle-avoidance cost for a particular twist $\mathbf{v}_i$ depends on the length of the assigned user plan as follows.

$$c_{oa}^{traj}(\mathbf{v}_i) = \left(\frac{1}{\Delta t_i + 1} \right)^{\beta_i^{travel}} \tag{5.26}$$

where the trajectory's length is measured by the effective travelling time, Δt_i, for traversing the trajectory at the twist $\mathbf{v}_i^{traj}$. The twist $\mathbf{v}_i^{traj}$ resembles the dynamic window twist $\mathbf{v}_i$ but is constrained to the assigned collision-free trajectory.

In obstacle-avoidance mode, the robot's linear speed should be maintained whereas the angular speed should be selected according to the available free space. Therefore,

$\mathbf{v}_i^{traj}$ on the assigned collision-free trajectory respects the linear component of the dynamic window twist $\mathbf{v}_i$ and adjusts the angular component to match the trajectory's curvature. This rule is invalid for trajectories turning on-the-spot. In this case, the angular speed of twist $\mathbf{v}_i$ is preserved, $\mathbf{v}_i^{traj} = (0, \text{sign}(\omega) \cdot |\omega_i|)$.

If the collision-free trajectory is parametrised by $(v, \omega, \Delta t)$ (cf. equation 5.2), the effective travelling time at twist $\mathbf{v}_i^{traj}$ equals:

$$\Delta t_i = \begin{cases} \frac{|v| \cdot \Delta t}{|v_i|} & v \neq 0, v_i \neq 0 : \text{circular arc } \boldsymbol{v}_i \rightarrow \text{circular arc} \\ \frac{|v| \cdot \Delta t}{\epsilon = 10^{-4}} & v \neq 0, \omega = 0, v_i = 0 : \text{pure turn } \boldsymbol{v}_i \rightarrow \text{straight line} \\ \frac{|\omega| \cdot \Delta t}{|\omega_i|} & v = 0, \omega \neq 0, v_i \neq 0, \omega_i \neq 0 : \text{non-zero } \boldsymbol{v}_i \rightarrow \text{pure turn} \\ \frac{|\omega| \cdot \Delta t}{|\omega_i|} & \omega \neq 0, v_i = 0, \omega_i \neq 0 : \text{pure turn } \boldsymbol{v}_i \rightarrow \text{circular arc} \\ \frac{|\omega| \cdot \Delta t}{\epsilon = 10^{-4}} & v = 0, \omega \neq 0, v_i \neq 0, \omega_i = 0 : \text{straight line } \boldsymbol{v}_i \rightarrow \text{pure turn} \\ \frac{|\omega| \cdot \Delta t}{\epsilon = 10^{-4}} & \omega \neq 0, v_i = 0, \omega_i = 0 : \text{origin } \boldsymbol{v}_i \rightarrow \text{pure turn} \end{cases} \tag{5.27}$$

In equation 5.26, the exponent β_i^{travel} influences the shape of the obstacle-cost. This cost should be more discriminative in the lowest interval of trajectory lengths when obstacle-avoidance truly matters. For longer trajectory lengths, the cost might be less differentiating. By design, the cost equals 0.1 for a fraction γ of the maximally possible travelling time, $\Delta t_i^{\max}$. This leads to an exponent β_i^{travel} of:

$$\beta_i^{travel} = \frac{\ln 10}{\ln\left(\gamma \cdot \Delta t_i^{\max} + 1\right)} \tag{5.28}$$

The choice of 0.1 as threshold is related to the cost's high variation in the interval $[0.1, 1]$. Effective travelling times in the range $[0, \gamma \cdot \Delta t_i^{\max}]$ are then mapped to the high variation interval $[0.1, 1]$, thus paying tribute to the importance of discriminating shorter trajectories for obstacle-avoidance.

Figure 5.13 shows the shape of the cost in equation 5.26 as a function of normalised travelling times and all possible twists $(v_i, \omega_i)_{traj}$ compatible with a collision-free trajectory. The normalised travelling times represent the trajectory's length. In this case the trajectory corresponds to a pure turn on the spot, i.e. $\left(v = 0\frac{m}{s}, \omega = 1.4142\frac{rad}{s}\right)$. According to equation 5.27, the effective travelling times are $\frac{|\omega| \cdot \Delta t}{|\omega_i|}$ and ω_i varies over the whole range of possible robot velocities.

The discontinuity in the cost plot for $\omega_i = 0\ rad/s$ stems from using a threshold value $\epsilon = 10^{-4}\ rad/s$ in the travelling time calculation instead of zero angular speed (cf. 5.27). As can be appreciated in figure 5.13, the cost monotonically decreases

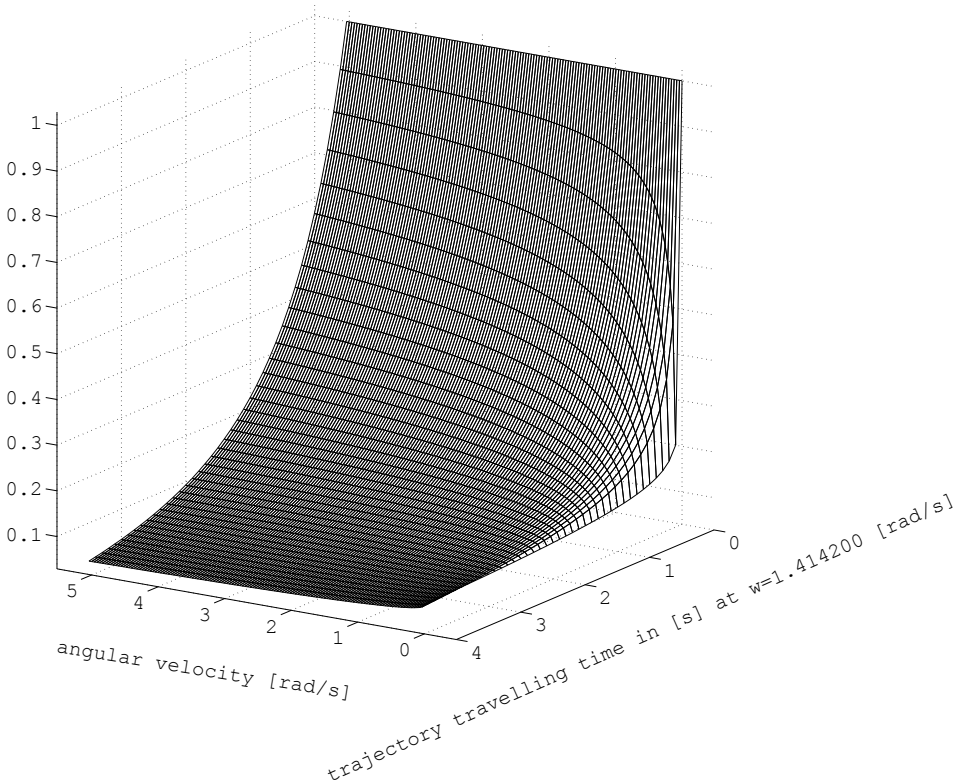


FIGURE 5.13: Shape of the obstacle avoidance cost

in function of the travelling time. The decrease is uniform for all possible robot velocities ω_i. It could be argued that the cost's decrease should change in function of the speed in order to punish shorter trajectories at high speeds. We have opted to ignore this dependency in the obstacle-avoidance cost and to consider it instead in the collision term, $c_{coll}(\mathbf{v}_i)$, as will be explained further on.

The obstacle-avoidance cost in equation 5.26 expresses a preference for longer collision-free trajectories. However, due to uncertainty in estimating the robot's position or uncertainty in executing motor commands, it is also interesting to select paths with higher clearance from obstacles. Therefore, an obstacle-clearance term is added to the obstacle-avoidance cost with the same form as the trajectory length term in equation 5.26. This leaves the overall cost as:

$$c_{oa}(\mathbf{v}_i) = \left(\frac{1}{\Delta t_i + 1}\right)^{\beta_i^{travel}} + \left(\frac{1}{clearance + 1}\right)^{\beta_{clearance}} \tag{5.29}$$

The obstacle clearance is defined as the average collision-free trajectory length of all trajectories in a neighbourhood of the current twist $\mathbf{v}_i$. Neighbourhoods $N(\mathbf{v}_i)$ in velocity space comprise a constant number of collision-free trajectories surrounding a particular twist.

$$clearance = \frac{1}{n_i} \cdot \sum_{j=1}^{n_i} \Delta t_j^{travel}, j \in N(\mathbf{v}_i) \tag{5.30a}$$

$$\beta_{clearance} = \frac{\ln 10}{\ln \left(\gamma \cdot clearance^{\max} + 1\right)} \tag{5.30b}$$

The same considerations described for the exponent of the trajectory related term, β_i^{travel}, apply to the clearance exponent, $\beta_{clearance}$, in equation 5.30.

The current obstacle-avoidance cost merely considers available free space to decide among moving directions. All twists assigned to the same collision-free trajectory are equally valid. In order to establish a preference among twists, the obstacle-avoidance cost is rescaled by the proximity to the user's linear velocity (equation 5.31).

$$c'_{oa}(\mathbf{v}_i) = c_{oa}(\mathbf{v}_i) \cdot (1 + |v_{user} - v_i|) \tag{5.31}$$

Additionally, the robot should circumvent objects in the direction selected by the user, i.e. it should avoid objects driving on the right side if the user is pointing right with her joystick. This is achieved by increasing the obstacle-avoidance cost of twists yielding a turn in the wrong direction.

Figure 5.14 presents the obstacle-avoidance algorithm in action. In the left part of the figure the current dynamic window and the cost for obstacle-avoidance are displayed. Collision-free trajectories are indicated by solid lines and dashed lines mark decision boundaries in velocity space. All twists within two dashed boundaries are assigned to the solid line contained in between. This assignment preserves the linear velocity, as explained previously. The right part of the figure shows the robot in the environment together with the collision-free trajectories matching the solid lines on the dynamic window.

The optimal obstacle-avoidance trajectory indicated as a dashed line in figure 5.14 is clearly the longest with the highest clearance. However, the overall optimal trajectory, which considers also the user cost c_{user} in equation 5.25, is among the longest but with smaller clearance. The effect of the user cost is to pull the global optimum in the dynamic window closer towards the user's input. In this example the cost for collisions at high speed, c_{coll}, is zero for all dynamically feasible twists since the robot is capable of stopping in time (cf. section 5.1.2).

Figure 5.15 compares the obstacle-avoidance cost with different user related costs for the same situation depicted in figure 5.14. Two orthogonal arrows mark the centre of the current dynamic window. User costs favour twists on the middle left of the dynamic window whereas the obstacle-avoidance cost is lower to the right of

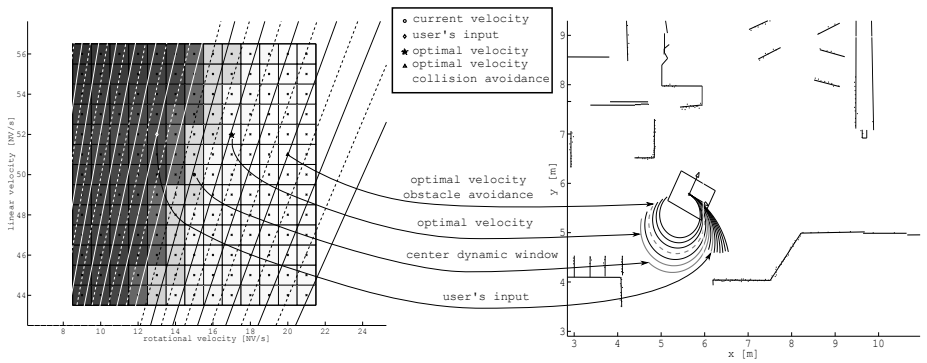


FIGURE 5.14: Cost for obstacle avoidance

the centre. Please note that in this case left and right side of the dynamic window do not correspond to avoiding obstacles to the left or to the right. Avoiding to the left or to the right is dynamically feasible only if the dynamic window includes the trajectory $\omega = 0\frac{rad}{s}$. The precedence between user and obstacle-avoidance costs is determined by the mixing weights α_{user} and α_{oa} in equation 5.14. Experimentally it was found that a ratio of $\frac{\alpha_{user}}{\alpha_{oa}} = 0.75$ showed good performance in balancing obstacle-avoidance and user related costs.

In order to ensure the user's safety, the robot punishes collisions at high speeds by means of the collision cost, c_{coll}. A collision at high speed occurs if the robot is unable to stop after processing motion commands at the user's reaction time (please refer to section 5.1.2 for further details). In case such a collision is inevitable, the cost c_{coll} increases with the robot's speed (cf. equation 5.32).

$$c_{coll}(\mathbf{v}_i) = \begin{cases} \max\left\{\frac{|v_i|-v_{th}}{v_{max}-v_{th}}, \frac{|\omega_i|-\omega_{th}}{\omega_{max}-\omega_{th}}\right\} & \left\{\begin{matrix} v_{th} < |v_i| < v_{max} \\ \omega_{th} < |\omega_i| < \omega_{max} \end{matrix}\right\} \\ 0 & \text{otherwise} \end{cases} \tag{5.32}$$

Assuming a linear collision cost as in equation 5.32, it is possible to derive constraints on the mixing weight, α_{coll}, in order to ensure that this cost overrules both the obstacle-avoidance cost and any user related cost. Following a similar reasoning as in the collision-avoidance section (sec. 5.2.1), the mixing weight α_{coll} should verify the next condition:

$$\alpha_{coll} > [\alpha_{user} + \alpha_{oa}] \cdot \max\left\{\frac{v_{\max} - v_{th}}{\Delta v_{DW}^{min}}, \frac{\omega_{\max} - \omega_{th}}{\Delta \omega_{DW}^{min}}\right\} \tag{5.33}$$

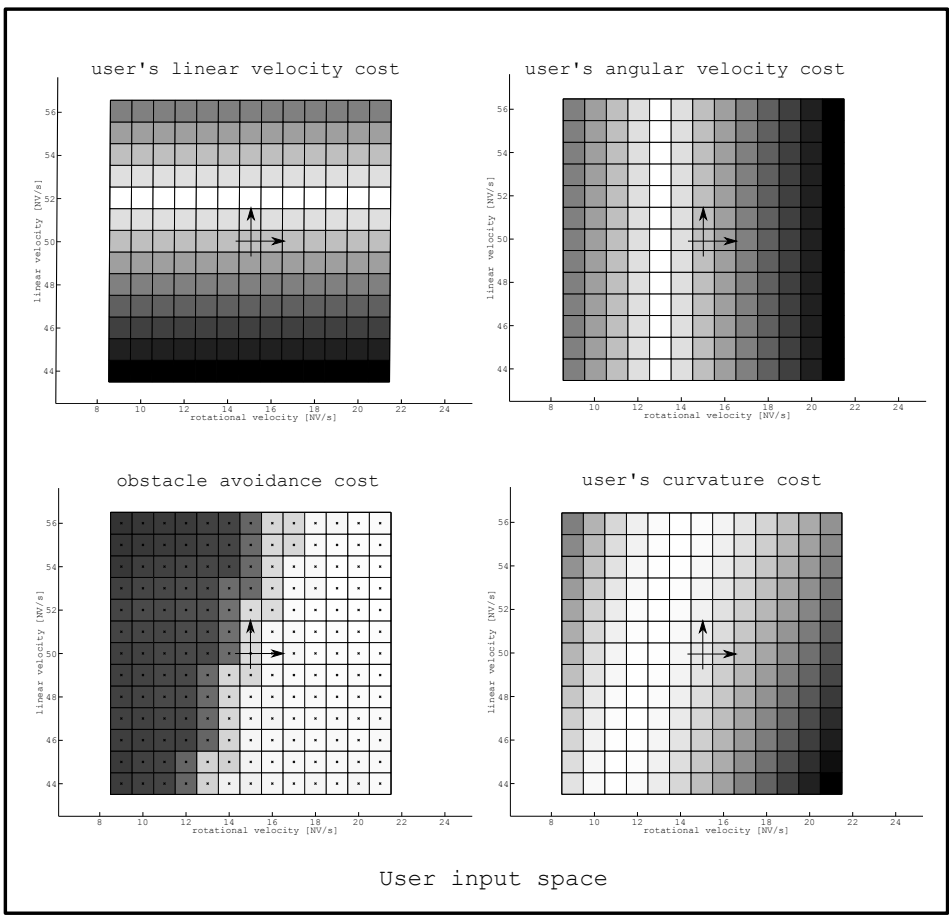


FIGURE 5.15: Cost for obstacle avoidance versus user costs.

5.3 Maximum-A-Posteriori (MAP) shared control

This section exemplifies the potential of shared control when considering the user's plan. The main objective of this shared control algorithm is to assist the user only in situations where assistance is really required. The user should always feel in control of the wheelchair, which in turn should behave as the user expects. Another necessary requirement related to the fact that a person is transported by the wheelchair is that navigation assistance should be safe and comfortable.

In order to achieve such intuitive human-robot navigation, this shared control algorithm estimates the plan of the user according to the procedure described in section 3.2.2. Based on the inherently uncertain plan estimate a decision is made about the assistance needs of the user, which the algorithm translates into a degree of control awarded to the user for a given situation. The degree of control determines how user inputs are merged with safe navigation commands of the shared controller.

Furthermore, the decision making process must be adapted to the driving style and abilities of each particular user in order to avoid frustrating her with wrong assistance. Additionally, the user should always feel in control of the wheelchair and be able to communicate her disagreement in case the robot behaves unexpectedly. In the latter case, the user should be able to override wrong navigation decisions of the robot.

We will show the benefits of shared control on a case study of a user with weak directional steering. This user has already been introduced in section 4.3.2, where an analytic user model for her specific driving disability was proposed. The interest of this particular driving limitation resides in its challenging requirements. Considerable information is removed from the user's input. Therefore, we will be able to demonstrate the benefits our framework for this driving disability, if the user is able to reach poses in the environment (read intentions) lying at her left when she intents to do so.

Please note that neither collision- nor obstacle-avoidance would be appropriate navigation assistance algorithms for this user since they completely disregard the user's driving style. This user would be unable to reach goal locations at her right with collision or obstacle avoidance because she would never be able to point her joystick in the right direction.

5.3.1 Safe navigation based on the user's plan

In order to assist the user the robot needs to devise a shared control strategy, which generates robot actions $\boldsymbol{a}_k$. Ideally, the robot would know the true plan of its user by observing her inputs combined with knowledge about her driving style. Then, robot actions $\boldsymbol{a}_k$ would simply translate the user's plan into a safe trajectory. Unfortunately, user interfaces only provide limited information about the user's

true plan. Therefore, the robot needs to infer what the user's plan might be before being able to provide navigation assistance.

Section 3.2.2 presented a probabilistic framework for recognising user plans, which estimates a posterior probability distribution over possible user plans, $bel(\boldsymbol{i}_k)$. This posterior summarises the robot's knowledge about the driving intentions of its user at time k. The shared controller then only needs to chose a plan according to a certain criterion, the *decision making rule*.

Since $bel(\boldsymbol{i}_k)$ summarises all knowledge about the user's plans, a possible inference of the user's plan could be the maximum of the posterior distribution. Let this plan be called the Maximum a Posteriori (MAP) plan and be denoted as $\boldsymbol{i}_k^{MAP}$.

$$\boldsymbol{i}_k^{MAP} = \arg\max_{\boldsymbol{i}_k} bel(\boldsymbol{i}_k) = \arg\max_{\boldsymbol{i}_k} p_{post}\big(\boldsymbol{i}_k \mid \boldsymbol{x}_{0:k}, \boldsymbol{u}_{0:k}, \boldsymbol{z}_{0:k}, \boldsymbol{a}_{1:k}\big) \tag{5.34}$$

The robot chooses $\boldsymbol{i}_k^{MAP}$ as an estimate of the user's true plan since at time k it is the plan for which it has accumulated most evidence. However, the $\boldsymbol{i}_k^{MAP}$ might not necessarily be the optimal plan in terms of driving comfort and its uniqueness is not guaranteed. Therefore, the robot needs an adaptive decision making scheme to break ties, while ensuring that the user is granted as much control as possible. We will discuss these issues in the next section.

The remainder of this section describes how the MAP plan $\boldsymbol{i}_k^{MAP}$ is converted into safe robot actions $\boldsymbol{a}$. This plan already enforces the robot's kinematics and is also collision-free. The robot's dynamics are considered within a global dynamic window approach.

In the experimental section we will consider only global user plans $\boldsymbol{i}_k$ because the user model for the user with weak directional steering is defined for these trajectories. However, tracking user plan $\boldsymbol{i}_k^{MAP}$ as described next is equally transferable to local plans.

The global dynamic window algorithm assigns costs to all reachable velocities around the current velocity of the platform. The cost of each velocity depends on the path tracking behaviour, on the distance to the actual user input and on the likelihood to cause a collision if the velocity is executed. Formally:

$$cost(\mathbf{v}_i) = \alpha_{user} \cdot c_{user}(\mathbf{v}_i) + \alpha_{grad} \cdot c_{grad}(\mathbf{v}_i) + \alpha_{coll} \cdot c_{coll}(\mathbf{v}_i) \tag{5.35}$$

The gradient cost $c_{grad}(\mathbf{v}_i)$ represents how well a given dynamic window velocity tracks the MAP path $\boldsymbol{i}_k^{MAP}$. It is calculated by integrating $\mathbf{v}_i$ for a certain time interval ΔT_{DW}. The gradient cost is directly proportional to the difference in position $(\Delta x, \Delta y)$ and orientation $(\Delta\theta)$ from the path to be tracked according to

the following expression:

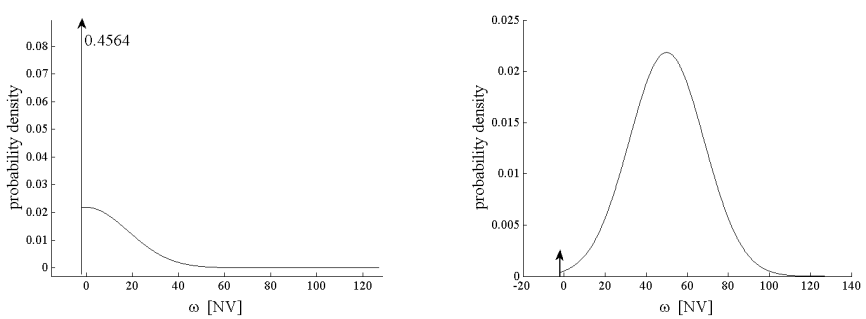
$$c_{grad}(\omega_i, v_i) = \frac{\sqrt{(\Delta x)^2 + (\Delta y)^2}}{\Delta L_{max}} + \beta_1 \cdot \left(\frac{1 - cos(\Delta\theta)}{2} \right)^{\beta_2} \tag{5.36}$$

where ΔL_{max} is the maximal linear displacement on the trajectory for ΔT_{DW}. β_1 and β_2 allow to control the weight and sharpness respectively of the angular displacement in the gradient cost function. The user cost c_{user} in eq. 5.35 is proportional to the Euclidean distance between the dynamic window velocity, $\mathbf{v}_i$, and the actual user input, $\boldsymbol{u}_k$. The collision cost $c_{collision}$ punishes velocities leading to a collision if executed during a certain time interval.

5.3.2 Adaptive decision making

The global dynamic window approach presented in the preceding section offers the means to merge the input of the user with the velocities needed to track safe trajectories. In order to avoid frustrating the user, the weights α_{grad}, α_{user} and $\alpha_{collision}$ in equation 5.35 cannot be held constant. If the user is able to execute a certain manoeuvre, full control should be granted to her. On the contrary, if the user experiences difficulties, she should be assisted in a safe manner. Therefore, it is mandatory to vary the weights in equation 5.35 continuously according to the estimated degree of control. If the user disagrees with the behaviour of the robotic wheelchair, she should be able to override the output of the controller.

In section 4.3.2 we derived a user model for a user with weak directional steering. The user model for user plans requiring a steering signal slightly to the left or forward and to the right were shown. Figure 5.16 repeats these user modelling results since they determine the adaptive decision making rule.



(a) Steering forward or slightly to the left.

(b) Steering to the right.

FIGURE 5.16: User model for a user with a weak left steering signal.

In figure 5.16 it can be observed that the user model tends to be more peaked at the threshold angular velocity, ω_{th}, the further the predicted intentional velocity ω^p lies to the left. The conditioned entropy, $H\left(\omega^{dis} \mid \omega^p = \hat{\omega}^p\right)$ captures precisely this observed phenomenon (cf. Fig. 5.17) and, therefore, is employed to decide on the degree of control granted to the user.

$$H\left(\omega^{dis} \mid \omega^p = \hat{\omega}^p\right) = -\int p_{user}\left(\omega_k^{dis}|\boldsymbol{i}_k\right) \cdot \ln\left(p_{user}\left(\omega_k^{dis}|\boldsymbol{i}_k\right)\right) d\omega_k^{dis} \qquad (5.37)$$

In the proposed decision criteria the user has less control when her signals are close to the threshold angular velocity ω_{th}. This fact is justified since the user cannot steer to the left. Thus, if the wheelchair steers to the left, the user could compensate for an incorrect action by steering back to the right. Consequently, steering to the left has a higher informative value for the proposed driving disability.

The entropy of the user model could also be employed to estimate some part of the degree of control granted to the user for other types of driving disabilities. The entropy of the user model measures the degree of uncertainty of the user model, i.e. of the confidence in the present user signals.

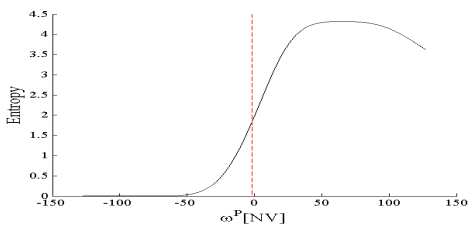


FIGURE 5.17: Conditioned entropy $H\left(\omega^{dis} \mid \omega^p = \hat{\omega}^p\right)$ depending on the position of the predicted intentional velocity $\hat{\omega}^P$.

However, the conditioned entropy of the user model is not enough to decide on the level of control granted to the user. It is also necessary to incorporate the confidence of the shared controller in its uncertain user plan estimates. Tracking wrong intentions might frustrate the user. Therefore, an additional criterion is introduced related to the percentage of uncertainty on the plan estimates. Such uncertainty is measured as the entropy of the posterior distribution $bel(\boldsymbol{i}_k)$ with respect to the entropy of the uniform distribution, which denotes the case of maximal uncertainty. In order to avoid frequent mode switches, the degree of autonomy is smoothed with a running average of N samples.

Should the user disagree with the provided assistance, then she can override the output of the shared controller by releasing the joystick. This is detected by the system and full control is granted back to the user in order to disambiguate the situation. As the degree of control is smoothed, granting full control corresponds with initializing all the values of the running average to 100.

The adaptive decision making rule for the assigned degree of user control (DC) is summarized in eq. 5.38:

$$DC = \begin{cases} avg \left[\begin{array}{c} \gamma_{user} \cdot H\left(\omega^{dis} \mid \omega^{p} = \hat{\omega}^{p}\right) + \\ \gamma_{cont} \cdot H_k\left(\boldsymbol{i}_k | \boldsymbol{u}_k, \boldsymbol{\mathcal{H}}_{0:k}\right) \end{array} \right] & \boldsymbol{u}^{dis} \neq \mathbf{0} \\ initialize_avg\,[100] & \boldsymbol{u}^{dis} = \mathbf{0} \end{cases} \tag{5.38}$$

where, γ_{user} and γ_{cont} allow to balance the influence of both terms. A good compromise for the weights γ is $\gamma_{user} = \gamma_{cont} = 0.5$, showing good experimental performance. The weights of the dynamic window (cf. eq. 5.35) are adjusted according to the estimated degree of control at each time step, i.e. $\alpha_{user} = 100 - \alpha_{grad} = DC$.

5.3.3 Experiments

The proposed approach for plan recognition and shared control was validated with K.U.Leuven's robotic wheelchair Sharioto (Figure 5.18) in the test environment displayed in Fig. 5.19. Several a priori intents have been placed in locations of potential interest to the user.

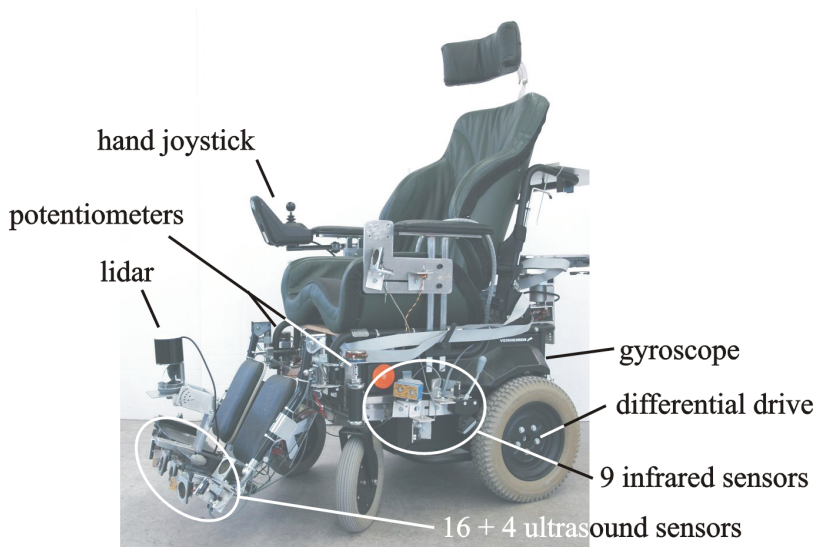


FIGURE 5.18: K.U.Leuven's robotic wheelchair *Sharioto* can be equipped with a hand joystick or a chin joystick. The wheelchair contains 9 infrared sensors, 16 ultrasound sensors with dead zone, 4 ultrasound sensors without dead zone, 2 potentiometers for measuring the orientation of the front castor wheels, 1 lidar (light detection and ranging sensor), and 1 gyroscope for rotational velocity estimation.

In this environment, a user with a weak left signal[4] was asked to perform manoeuvres involving turns to the left. The maximal turning velocity the user could achieve for left turns was set to $\omega_{th} = 0.0812\ rad/s$, which is too low to turn on the spot with the motor controller of our electrical wheelchair Sharioto.

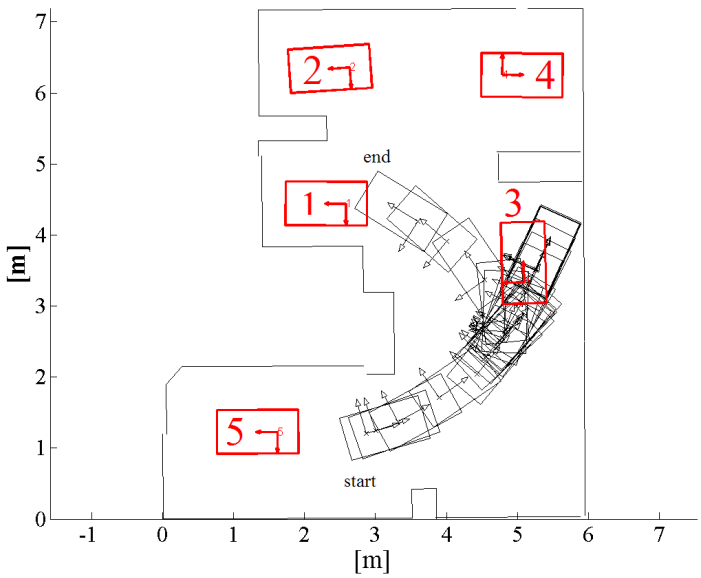


FIGURE 5.19: Manoeuvre executed in a test environment. Possible intentions are displayed in red.

The experiment began at the pose marked as start and finished when the user reached the pose denoted end. With the imposed steering disability the user could not have performed such a manoeuvre. The shared controller detected the plan of the user and assisted whenever assistance was needed. In figure 5.20 it is possible to observe the evolution of the posterior probability over user plans $bel(\boldsymbol{i}_k)$ for the executed manoeuvre. The red circles denote the maximum of the posterior, determining which path should be tracked at each time instant, and the solid blue line shows the true user intent.

Figure 5.21 displays the evolution of the degree of control granted to the user during the complete manoeuvre. Higher values imply the user is in control. In case the user disagrees with the wheelchair's behaviour, she can communicate her disagreement by releasing the joystick. This explicit communication channel can override the behaviour of the robot because it resets the decision rule.

In the first part of the manoeuvre, the user reaches intention 3 (time steps $1 - 86$). This is correctly detected as $bel(\boldsymbol{i}_k)$ converges to plan 3. Once the user has reached

[4]The weak left signal was simulated by distorting the original input of an able-bodied user according to the mapping specified in eq. 4.12b.

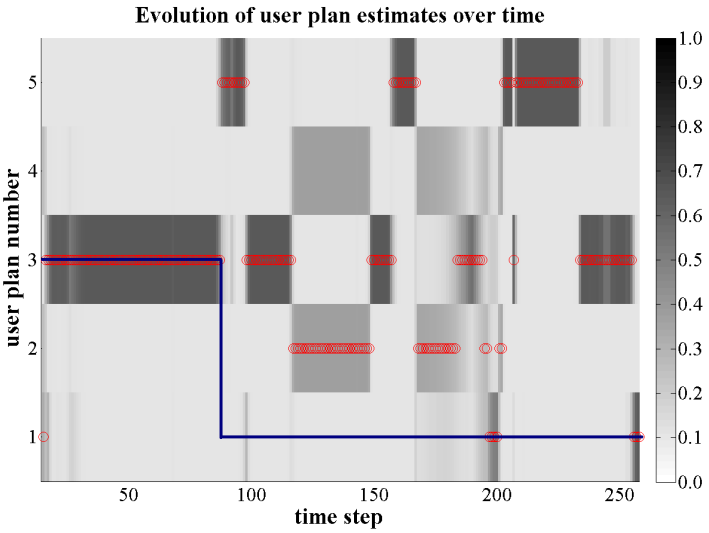


FIGURE 5.20: Evolution of the posterior probability $bel(\boldsymbol{i}_k)$ during the manoeuvre. The circles denote the maximum of $bel(\boldsymbol{i}_k)$, whereas the solid blue line denotes the true user plan. Darker colours indicate a higher value of the posterior.

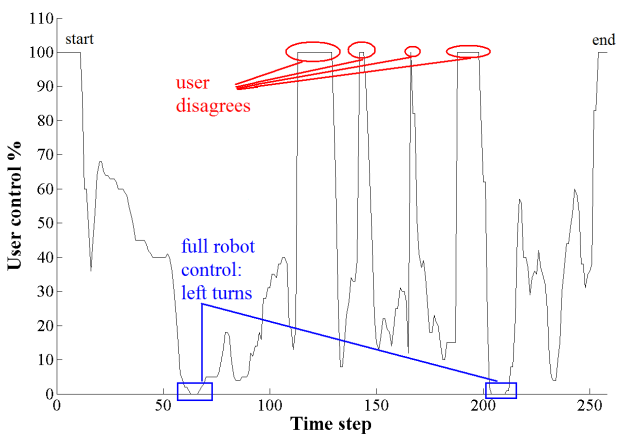


FIGURE 5.21: Degree of control granted to the user during the manoeuvre. The higher the value, the more control is granted to the user. The user can override the decision of the shared controller by releasing the joystick, after which the degree of user control will automatically increase to 100%.

end pose 3, she will try to get out of that position. Given that she cannot steer to the left and turning to the right is not possible, all the user can do is to drive backwards (time steps $86 - 98$).

Then, the user probes if the shared controller already detects that she intents to turn (time steps $98 - 117$). Plan 3 maintains the highest probability and the wheelchair drives again forward. User plans 2 and 4 have a higher posterior probability during time steps $113 - 136$, which causes a slight turning of the wheelchair to the left. In order to communicate her disagreement with the observed action, since the wheelchair is not turning enough, the user releases the joystick (time steps $136 - 153$). This is detected and the user is granted a higher degree of control (cf. figure 5.21).

During time steps $153 - 177$ the user drives backwards again, this time for a longer distance. Afterwards, the user drives forward with the plan of driving to pose 1. User plans 2 and 4 have the highest probability (time steps $168 - 183$), matching the user plan of driving forward. Not wanting to be trapped again in the corner of pose 3, the user releases the joystick showing disagreement. Note that plan 3 has the highest probability during time steps $183 - 195$. The wheelchair stops and the degree of user control is immediately increased to its maximal value. Rapid plan changes cause the wheelchair to initiate turning. For this manoeuvre, the user is granted no control at the beginning (cf. figure 5.21 situation $203 - 212$). During turning, the user plan with highest probability is user plan 5 (time steps $202 - 232$).

At almost the end of the manoeuvre (time steps $237 - 254$), user plan 3 has the highest probability, in accordance with the input given by the user. The user is steering backwards in order to brake the wheelchair. Once the wheelchair is driving slowly, the user pushes the joystick slightly forward to reach end pose 1.

Observing the evolution of $bel(\boldsymbol{i}_k)$ in figure 5.20 we can deduce that the plan recognition process does not always converge to the true intent. This will not cause any unpredictable behaviour of the wheelchair, as long as the maximum probability path corresponds, for a certain interval, to the path the user has in mind.

5.3.4 Discussion

The preceding sections present a shared control algorithm, which tracks the maximum probability fine-motion paths estimated during plan recognition. The shared controller uses an adaptive decision scheme, which explicitly incorporates the uncertainty on the plan estimates when merging the input of the user with the output of the path tracking algorithm. First experimental results are reported on a real wheelchair for a user with a weak left steering signal. This user is able to reach poses in the environment, which would be inaccessible to her without navigation assistance.

The simple experiment already reveals two fundamental properties of human-robot navigation. Firstly, navigation assistance is enhanced by estimating the user's plan in such a way that considers the particular driving style of the user. Furthermore, there is a need for bi-directional communication between human and robot, in

case the human disagrees with the navigation assistance. This bi-directional communication allows to *share the control* over the robot effectively.

5.4 Conclusions

This chapter presented a general framework to share the control over a robotic wheelchair with its user. Shared control is the action-selection component of the navigation assistance architecture. The robot must choose actions, which are physically executable. Otherwise, the robot might fail in exactly the same situations users find troublesome. Further, the robot must ensure the safety of its user by considering obstacles in the environment. And lastly, intuitive control of the robot is possible only if the robot reasons about its user's navigation plans in such a way that considers the user's abilities. Conversely, the user will be frustrated if the robot consistently disregards her intentions and driving style.

In order to address these requirements, our shared control framework is completely integrated with the plan recognition component described in section 3.2. The basic building block in the framework are a set of collision-free paths, which represent all potential user plans. These paths are kinematically feasible and action selection during shared control respects the robot's dynamics through the Dynamic Window Approach (DWA).

We have designed and implemented three shared control algorithms in the DWA, where the user gradually cedes more control to the robot. The first two algorithms, *collision-avoidance* and *obstacle-avoidance* have a long tradition in mobile robotics, sometimes meaning different things depending on the context. Our implementation restricted these algorithms to human-centred navigation functions, which respect the robot's kinematics and dynamics. We have shown how *collision-avoidance* and *obstacle-avoidance* are special cases within our navigation assistance architecture. These shared control schemes appeal to users who want to drive fast and safely, without loosing too much control over the robot.

The third algorithm, *Maximum a Posteriori (MAP)* shared control, helps the user to navigate by estimating the user's plan. In MAP shared control, the robot considers the maximum of the posterior distribution as the most probable plan of the user, the MAP plan. This is the plan for which the robot has accumulated most evidence. An important requirement for MAP shared control is to involve the user as much as possible in navigation, in order to keep the user physically and mentally fit. Therefore, the robot also estimates the amount of control granted to the user online and helps only to the required extent.

We have demonstrated the performance of MAP shared control for a user who could not steer left. Thanks to navigation assistance, this user was able to reach intentions at her left when intended, despite her driving disability. Please note

that collision- or obstacle-avoidance would have failed for this user, because this user was unable to point the wheelchair in the right direction for left turns.

The MAP experiments revealed two fundamental properties of human-robot navigation. Firstly, we have demonstrated that it is beneficial for navigation assistance to estimate the user's plans by considering her individual driving abilities. Secondly, in order to share the control effectively, the user must be able to communicate disagreement through a bi-directional communication channel.

Chapter 6

Conclusions

"Caminante, son tus huellas
el camino, y nada más;
caminante, no hay camino:
se hace camino al andar.
Al andar se hace camino,
y al volver la vista atrás
se ve la senda que nunca
se ha de volver a pisar.
Caminante, no hay camino,
sino estelas en la mar."

Wanderer, your footsteps are
the path, and nothing more;
wanderer, there is no path:
one makes the path by walking.
By walking one makes a path,
and when looking back
one sees the track which never
will be trodden again.
Wanderer, there is no path,
only wakes upon the sea.

Antonio Machado (⋆1875 – †1939)
Proverbios y Cantares. XXIX. Campos de Castilla (Machado, 1912)

As the poem of Machado suggests, mobility is linked intrinsically to our human condition. There is no predefined path but the one made by advancing through life. Therefore, mobility is a very important aspect in the feeling of independence and consequently has a great impact on a person's self-esteem.

Unfortunately, a large group of individuals is constrained in their mobility. Electric wheelchairs are a possible way to recover mobility. However, not all potential users are able to control a wheelchair properly. In some cases, electric wheelchairs are not prescribed due to the user's lack in motor or cognitive skills. Even seemingly easy manoeuvres such as driving through a door, following a corridor, or avoiding obstacles are experienced as difficult, time-consuming and (therefore) annoying. The relatively large size of wheelchairs compared to the dimensions of places in people's homes or in hospitals, makes navigation even more difficult. Clearly, there

is a need for navigation assistance in order to enlarge the group of users benefiting from wheeled mobility.

6.1 Contributions

This work proposes a framework for human-robot navigation. The human shares part of her autonomy with the robot in return of safer driving. The goal of the robot is to provide user-specific navigation assistance for difficult or tiring driving situations, while leaving the user as much in control as possible. Granting the maximal level of control to the user is beneficial to enhance the user's self-esteem, but also to ensure the user remains active to the best of her abilities.

The user will perceive the robot as intuitive if she has the impression that the robot understands her intentions and only provides the necessary amount of driving assistance. Only then will she be prepared to share part of her autonomy. Therefore, the robot needs to estimate the user's navigation plans, which we modelled as trajectories that the robot should execute. Based on the estimated trajectory, a robust decision can be made for navigation assistance. The robot's behaviour can be made inherently safe by taking only safe trajectories into account, i.e. trajectories that are collision-free and that respect the robot's kinematics and dynamics.

The framework for navigation assistance consists of two interrelated processes, recognising the user's plan and sharing the control over the robotic wheelchair. During plan recognition the robot considers the user's specific driving style in order to estimate the plans of its user. Further, the robot bases its shared control decisions on these estimated plans, which allows effective navigation assistance.

The following sections describe the contributions of this work in *plan recognition* and *shared control*, which rely on *probabilistic models of the user*'s driving style.

6.1.1 Plan recognition

The first contribution of this thesis is a formulation of human-robot navigation as a Dynamic Bayesian Network (DBN). *Plan recognition* is a subset of the complete state estimation the robot faces, who additionally needs to estimate its position with respect to an uncertain map. We have divided human-robot navigation into two sub-problems, simulataneous localisation and mapping (SLAM) and plan recognition. The rationale behind this division is to achieve a modular design of the robot's architecture and in order to simplify the solution to each problem.

We have presented a robust plan recognition approach, which considers the uncertainty in estimating user plans. Such uncertainty stems from two sources. On the one hand, there is inherent uncertainty related driving. For instance, the user might have misconceptions about the robot's behaviour or might drive imprecisely.

On the other hand, uncertainty also originates as a consequence of modelling errors. Therefore, a more robust approach to sharing the control with the user explicitly considers the level of uncertainty when taking assistance actions.

We have discussed the impact of modelling user plans as global or local trajectories. Temporal reasoning with global plans is simple, but achieving a dense set of plan hypotheses is more involved. The plan library needs to include all homotopically different plans. Conversely, local plans are beneficial for immediate navigation assistance, at the expense of increasing the complexity in temporal reasoning. Global references are lost for local plans, which would allow a straightforward initialisation of the prior for the next time step. Therefore, the major contribution in plan recognition is a procedure to reason about the temporal evolution of local user plans.

We proposed in section 3.2 a solution to local plan recognition in the human-robot DBN. The fundamental insight to solve the problem is that complete certainty is possible only through global knowledge. Therefore, we have proposed to employ the robot's future trajectory, as a source of global certainty. This trajectory is known after collecting the data thanks to the benefit of hindsight.

At every sampling time, the plan library $\boldsymbol{\mathcal{I}}_k$ contains a dense set of local trajectories. Each plan of $\boldsymbol{\mathcal{I}}_k$ is a potential future trajectory of the robot. Hence, it becomes possible to compare potential future paths with the actual path of the robot. We have analysed candidate trajectory metrics in order to formalise the notion of similarity between paths. For the selected metric, we have studied its statistical properties to shed light on the probabilistic structure of the problem. Unfortunately, the statistical properties of the metric alone are insufficient to define a solution to local plan recognition. We had to introduce additional domain knowledge to compute the solution. This knowledge imposes that a user plan is more likely to correspond to the robot's future trajectory the closer it lies to it.

We formalised the domain knowledge in a subjective *distance-likelihood*, which we combined with other sources of uncertain knowledge to calibrate the *robot-specific* component of plan recognition, i.e. the *plan process function*. The rationale behind the distance-likelihood is to reduce the amount of data needed during training of the models. Normally, it would be required to observe sufficient transitions between *all* possible user plans for *all* possible actions. The method we proposed reduces this amount of data because knowledge about observed transitions propagates to unobserved plan transitions. Further, it also allows to control the level of noise of the plan process function to avoid overconfident posterior distributions.

We demonstrated the validity of the approach on a simulated wheelchair with a discrete interface, as well as for a real wheelchair with a continuous joystick.

6.1.2 User modelling

The predictive power of plan recognition depends to a large extent on the accuracy of the driver's model. This *user model* is the cornerstone of our probabilistic plan recognition framework. It allows to adapt plan recognition to each user because it encapsulates how a particular driver transforms her mental plans into inputs to the robot.

Concerning user modelling, the main contribution of this thesis is the analysis of a real user's driving behaviour. During the experiments, the user had to maintain a very high level of concentration in order to fulfil a set of navigation tasks. Despite her best efforts, collisions with the environment still occurred. Therefore, this user could clearly benefit from a robotic wheelchair, which would assist in the most dangerous or tiring situations.

We have proposed a general user modelling technique based on *Gaussian Processes* ($\mathcal{GP}$), which can extract driving styles directly out of driving examples. We have justified the choice of this technique through an analysis of user modelling requirements, and addressed how specific choices of the $\mathcal{GP}$-user model fulfil these requirements. The $\mathcal{GP}$-user model calculates a probabilistic mapping between user plans and user inputs. Therefore, it can predict the user signal's likelihood given a certain plan.

We have collected driving examples of a spastic user navigating in a daily environment and calibrated a $\mathcal{GP}$-user model. First, we determined ground-truth with the same trajectory metric employed to derive the plan process function. During the experiments we provided no navigation assistance because the main objective was to study the user's driving behaviour. The actually executed trajectory matched the user's inputs to the robot. Therefore, it was possible to acquire a user model through ground-truth.

The $\mathcal{GP}$-user model is calibrated in a two-stage procedure. First, a full Gaussian Process is computed for the angular, $\mathcal{GP}_\omega$, and for the linear, $\mathcal{GP}_v$, components of the model. In this first stage, the hyper-parameters of the $\mathcal{GP}$-user model, such as the free parameters of the covariance function or the type of covariance function, are optimised via Automatic Relevance Determination. During the second stage of user modelling, we compute a sparse online approximations to the full Gaussian Processes. The second stage is incremental, i.e. data are presented sequentially. This makes the sparse approximation very appealing to analyse short-term fluctuations in driving. The full Gaussian Process with Automatic Relevance Determination, on the other hand, can be applied whenever larger scale variations of the user's driving style appear. The user's condition might worsen with time, but users could also improve through experience. In both cases fundamental changes of the user model might be necessary.

Encouraging results for plan recognition and user modelling have been achieved in estimating the spastic user's driving intention. During the largest part of the

experiments the robot could anticipate the user's plan and predict her future path.

In the discussion of the experiments we have also addressed the level of noise contained in the plan process function, which defines the plan recognition process. An overconfident plan process function is undesirable in case of convergence to a wrong plan. In order to increase robustness we have opted to maintain a certain amount of noise during plan recognition.

6.1.3 Shared control

The framework for shared control is the decision making component in the human-robot navigation architecture. An important requirement during navigation assistance, is that the robot's kinematic and dynamic constraints are considered. Otherwise, the robot might fail in the very same situations that users find troublesome. Since the robot reasons about collision-free user plans, the robot's kinematics are automatically considered. Further, action selection during shared control respects the robot's dynamics.

The basic algorithm for shared control is a human-centric implementation of the dynamic window approach. Traditionally, the dynamic window has been applied to fully autonomous robots as an elegant solution to the general trajectory planning problem. When sharing the control with a human, completely different requirements are imposed on the robot because a human is transported. The robot should respect the user's wishes, ensure her driving comfort, and occasionally allow collisions to interact with objects in the environment.

Human factors play a paramount role in the design of the shared controller. User characteristics such as the reaction time to unforeseen events or the information processing bandwidth must be considered by the robot. For instance, the robot checks if it has sufficient time to brake after possible delays in the reactions of the user. The bandwidth of the algorithm is also constrained by the user's bandwidth, since, in principle, shared control could operate at faster rates and cut the user out of the loop. This is undesirable in human-robot navigation.

Three shared control algorithms were proposed, each with varying levels of user autonomy. *Collision-avoidance* is the most basic level of navigation assistance, which ensures collisions are only possible at low speeds. *Obstacle-avoidance* reduces the user's autonomy slightly by helping her to steer away from obstacles. *Collision-* and *obstacle-avoidance* a generic terms for a myriad of mobile robotics algorithms. Our definition of these algorithms imposes that the robot's kinematics and dynamics are respected. Further, our implementation is human-centric, although not adapted to the particular driving style of a user. *Collision-* and *obstacle-avoidance* are mainly interesting for users who like to drive fast in cluttered environments, but do not want to cede too much control. It was shown that it is possible to implement these standard algorithms in the proposed human-robot navigation architecture.

They serve as a reference to compare other shared control algorithms, as well as a basic safety layer which ensures collisions are only possible at low speeds.

The third shared control algorithm is based on the best estimate of the user's plan given all available information to the robot up to a certain time. This algorithm is called Maximum A Posteriori (MAP) shared control, since it estimates the plan of the user as the maximum of the posterior probability over user plans $bel(\boldsymbol{i}_k)$. In order to demonstrate the benefits of MAP-shared control we presented a case study of assistance to a user with weak directional steering. This particular disability prevents the user from reaching destinations at her left. Collision- or obstacle-avoidance would prove insufficient because the user is incapable of generating commands in the correct direction.

The chosen case study allows important insights into the problem of human-robot navigation. Firstly, estimating user plans by considering the user's particular driving style proves invaluable to provide correct assistance. If the robot is also aware of the uncertainty about user plans, it can delay its decision about assistance until it is certain enough to avoid frustrating the user.

Furthermore, an important requirement is that the user should always feel in control despite sharing her autonomy with a robot. The user and robot need therefore a bi-directional communication channel by which the user can communicate her disagreement about incorrect navigation assistance. We have proposed a natural communication protocol over a channel of reduced information capacity, namely the standard joystick of the robot. By releasing the joystick, full control is granted back to the user. This strategy is the most efficient means of communication without additional control switches.

6.1.4 User's perspective on navigation assistance

An important contribution of the thesis is the global view of navigation assistance. We have analysed the process of *recognising user plans*, *modelling driving behaviour* and *sharing the control* over the wheelchair between human and robot. We have shown how these processes allow to assist disabled users to navigate a wheelchair safely. The main motivation for this approach is to develop robots, which can *collaborate effectively* with a *heterogeneous user group*, but at the same time are *intuitive* and *easy to control*. Hence, this view is driven by the robot.

In this section, we follow a complementary approach and we will look at the problem of human-robot collaboration from the user's perspective. Let us image that a user recently acquired a new robotic wheelchair. We will describe the necessary procedure to start operating this wheelchair, which implements our navigation assistance architecture.

Role of the wheelchair manufacturer. The manufacture must test and calibrate the robotic wheelchair before selling it to a new customer. As part of this procedure, the manufacturer trains the wheelchair-specific *plan process function* $p_{ppf}(\boldsymbol{i}_k \mid \boldsymbol{i}_{k-1}, \boldsymbol{a}_k, \boldsymbol{x}_k, \boldsymbol{M})$ that models the probabilistic transitions between user plans $\boldsymbol{i}$ in response to robot actions $\boldsymbol{a}$.

As explained in section 3.2 the plan process function is independent of a particular user. It reflects how the kinematics and dynamics of the robot influence changes in user plans. Further, the plan process function assumes perfect knowledge about the robot's position $\boldsymbol{x}_k$ and environment map $\boldsymbol{M}$. This implies the localisation and mapping modules of the robot have to be calibrated as well. As an optional step, the wheelchair can accept a-priori maps from the user. The impact of localisation and mapping on the plan process function is to adapt user plans $\boldsymbol{\mathcal{I}}$ to the environment. The plan process function is not parametrised over possible maps and environment positions.

Role of the therapist who trains the user. Before a novel user can start driving the robotic wheelchair, she must first undergo an extensive training by ergo-therapists. Such training occurs for standard electrical wheelchairs as well. The novice user needs to learn how to drive the wheelchair, explore the wheelchair's capabilities and exercise the most common manoeuvres. The ergo-therapist adjusts several parameters of the wheelchairs such as maximal accelerations and speeds according to the user's abilities. The conventional training procedure is mostly qualitative and lacks a clear measure of performance or progress.

An important contribution of this thesis is a scientific method to model the user's driving style through a probabilistic *user model*, $p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k)$. Since this model captures how a particular user transforms her plans into inputs for the robot, the task of the ergo-therapist is clearly defined. Besides teaching the user how to drive, the ergo-therapist must ensure that the user model accurately represents the user's driving style. In order to guarantee an accurate user model, the training procedure must include sufficient training examples for *all* combinations of user plans and user inputs.

Further, the plan library $\boldsymbol{\mathcal{I}}$ provides a dense set of examples the user should master because these trajectories are encountered in daily operation. The user model also provides a way to assess the user's driving performance. It is possible to measure the spread of observed user signals around the mean of the user model distribution for a reference trajectory. Such comparison is user-specific and allows to monitor progress if the user model changes appreciably.

Daily use and monitoring. Once the user knowns how to drive the wheelchair and the robotic wheelchair knows its user, the wheelchair is ready for normal operation.

The user adapts to the wheelchair during a life-long process beyond the initial training. Her driving style changes at two time scales. In the short term, the user

can be motivated and concentrated and, thus, drive better. It is also possible that the user becomes tired and, consequently, her driving worsens. On the longer term, the user might improve through experience, or the user's ailment might evolve and affect the user's driving abilities.

The $\mathcal{GP}$-user model we have proposed in this thesis is capable of tracking changes at two time scales. Over short periods of time, the online $\mathcal{GP}$-user model can incorporate new driving examples incrementally, if the new situations are innovative enough. In order to achieve this adaptation, the user's intention $\boldsymbol{i}_k$ must be known to the robot. For instance, the maximum of the posterior distribution, if the distribution has converged, could be considered the user's instantaneous intention. Comparisons of the current situation to similar instances in the same environment in the past, can also provide clues about the innovative character of a situation.

At the end of each day, while the wheelchair is charging, the robot could analyse the complete driving dataset to calibrate a new $\mathcal{GP}$-user model from scratch. The robot can then compare if the user's driving has changed over the course of a day and if this change is significant for longer-term trends. Additionally, the robot could send the database of user models to a therapist or a doctor in order to monitor the user's evolution.

Another source of information about the user is the performance of the complete navigation assistance system. For instance, frequent disagreements between user and robot can be either due to failures of the robot, due to errors of the user, or due to changes in the user's abilities. In either case, such disagreements are informative and should be quantised. In section 5.3.2 we proposed an adaptive decision rule, which estimates the degree of control granted to the user online. If this decision rule consistently shows disagreement between robot and user, an intervention becomes necessary. This intervention can be automated by recalibrating the user model. It might also involve a doctor, a therapist or an engineer, if it concerns the robot, to diagnose the source of the problem.

6.2 Future work

6.2.1 Plan recognition

Plan recognition in human-robot navigation is a field with still many unanswered research questions. In the following we list a few promising directions of future research.

The human operator, even in the reduced realm of human-robot navigation, is still a great unknown. We have presented a framework to recognise the user's plan by means of a limited set of plausible plan hypotheses. However, the question remains if any finite set can ever be complete under any circumstances. In the limit,

the robot would need to reason in a continuous, high-dimensional hypothesis space, which also includes any conceivable motion profile. Future work will therefore investigate plan recognition in a continuous hypothesis space.

Another interesting point of future research is to consider context, such as time of the day, during plan recognition. The user may behave differently depending on her level of fatigue or her level of concentration, which vary during the day. The question then is how to combine multiple user models and plan process functions and to detect when to switch between them.

When controlling the robot, the user maintains a mental model of robot behaviour. This model needs to be adapted if the robot's assistance strategy evolves to accommodate better to its user. The result is a gradual interaction of two learning systems. On the one hand, the robot is learning about its user and, on the other, the user learns about the robot. Future work will analyse how to detect changes in the mental model the user is maintaining about the robot in order to limit cases of mode confusion, i.e. when human and robot do not share a common understanding of a navigation task.

The robot is the complementary perspective to the problem of human-robot navigation. Improvements and future challenges also remain from the robotics side. Firstly, to achieve more robust human-robot navigation, it would be necessary to consider a user plan representation which includes localisation uncertainty. Plan recognition would then be solved in combination with SLAM. The robot should still be able to reason about the user's plan and offer navigation assistance despite its localisation uncertainty.

Section 3.2.4 discussed a set of soft constraints on the true user plan, which define a solution to the local plan process probability. It would be interesting to investigate if such constraints could also be devised for a smooth assignment of robot actions to user plans. This is particularly important in the case of continuous robot actions, cf. section 3.3.2.

We believe that the concept of intuitive human-robot interaction is also transferable to other domains besides navigation. If a robot is collaborating with a human it is imperative that both share a common understanding of the task at hand. This understanding can be improved if human and robot know each others' plans. The problem of estimating user plans from an interface that offers partial information is of general value. Changes are necessary only in the representation of user plans, in the translation of user plans into robot inputs, and, possibly, in the set of robot actions. We have shown that this knowledge is encapsulated in a user model and a plan process function, which might differ from these models in navigation. However, the framework to reason about user plans and to acquire such models remains valid. Therefore, we will will investigate how to extend plan recognition to other robots performing arbitrary rigid body motion in space.

Computational concerns will also be addressed in future research. We have presented a procedure based on Dynamic Bayesian networks to calibrate the plan process function. This procedure does not scale to large amounts of driving data, for instance the driving data gathered during a whole day. Therefore, we will need to devise computational techniques to process vast data sets sequentially, for instance, by segmenting data into partial trajectories and matching equivalent parts.

6.2.2 User modelling

The encouraging results in user modelling and plan recognition reported in chapter 4 open new opportunities of future research along the following lines.

Modelling The predictive power of our probabilistic framework for plan recognition and shared control depends to a large extent on the quality of the user model. Therefore, future work will investigate ways to reduce the user model's uncertainty while keeping its predictions accurate. The user model's uncertainty should also scale when increasing the density of plans. For instance, each plan could be tracked at different acceleration or deceleration profiles. Further, more complex manoeuvres could be added to the plan library as well.

Future work will try to asses the user's reaction time and user's bandwidth experimentally in order to determine the optimal frequency for navigation assistance.

In the future we will also perform experiments with a larger population and analyse how driving styles differ across individuals. The framework for plan recognition allows to make such comparisons quantitatively. Pre-processing of the user's input such as tremor reduction might be required for certain users.

Long-term evolution and gradual adaptation Future work will involve long-term experiments to monitor the evolution of driving performance. The goal is to answer a series of research questions, such as when it is necessary to change the user model, or what is the bandwidth of user model changes that still results in an intuitive and easy to control robot. Another interesting aspect to consider is how navigation assistance influences the user model. In the EU IST project (RADHAR, 2010) we will study this interaction as a process of *R*obotic *AD*aptation to *H*umans *A*dapting to *R*obots (RADHAR, 2010).

Benchmarking Benchmarking is imperative in human-robot-interaction in order to move "from an art to a science" (Guizzo and Mandelbaum, 2010), in the

words of Dr. Mandelbaum[1]. In the experimental section we have introduced a benchmark to measure the performance of our framework for plan recognition and user modelling by predicting the user's future path.

The theoretical appeal of the user model is that it can also be employed to evaluate shared control algorithms in a standardised way. A user model represents a particular steering disability derived from real driving data. In (RADHAR, 2010) we will therefore develop a 'virtual user', who generates input signals by sampling the user model distribution. These samples represent noisy projections of the ground-truth plan, and would probably lead to collisions with the environment. Shared control algorithms can then be compared in a series of standardised tasks to determine the extent in which collisions are avoided.

Computation From a computational point of view there is still room for improvement. The focus of the experimental section was to understand human driving performance, as well as to analyse the predictive capabilities of the framework. Computation of the user model is in the range of $0.4s$ for a plan library of 144 trajectories for two Gaussian Processes ($\mathcal{GP}_\omega$ and $\mathcal{GP}_v$) with 100 representative points. Half a second is probably in the range of the user's bandwidth. The goal of (RADHAR, 2010) is to achieve navigation assistance at $5Hz$. The user model was computed for all trajectories without consideration of the wheelchair's dynamics. Updating only dynamically feasible trajectories would probably reduce the computational cost. An additional optimisation would be to compute the user model's likelihood in parallel with specialised hardware, such as a graphical co-processor.

6.2.3 Shared control

The chapter on shared control presented the basic framework for navigation assistance. Many avenues of research are still open in human-robot navigation.

Interaction of two learning systems. Human-robot navigation implies the interaction of two learning systems. The robot is learning about its operator while the user is learning how to control the robot. Both actors continuously adapt to each other. Therefore, future work will study shared control algorithms that model this interaction of two learning entities and provide customised assistance.

[1]Dr. Mandelbaum introduced the four-year DARPA Autonomous Robotic Manipulation (ARM) program, with the goal of enabling robots to autonomously perform complex tasks with humans providing only high-level guidance. Dr. Mandelbaum identifies in the interview (Guizzo and Mandelbaum, 2010) two aspects which differentiate "an art from a science", and which form an integral part of DARPA's ARM program. The first is *repeatability*. Research results should be reproducible, which is problematic with different robotic platforms. DARPA's ARM program provides therefore a standardised hardware platform to develop control software. The second aspect is *benchmarking*, or measured performance. The aim of the ARM program is to solve a set of standardised benchmark tasks.

An important aspect of this interaction is the bi-directional communication between user and robot. Navigation is intuitive only if both share a common understanding of each others' plans. We believe that haptic interfaces have the potential to address the communication needs of human and robot during navigation. The robot can communicate its intentions back to the user through a faster control loop. The user can, in turn, override incorrect navigation actions even before the robot moves. In future work, robot actions will be extended by haptic commands in addition to navigation actions. Appropriate decision rules will have to be developed, which prioritise both kind of actions.

Advanced Decision Making. MAP-shared control is only one possibility to convert estimated user plans into assistive actions. Other, more advanced, frameworks for decision making under uncertainty are conceivable. For instance, Partially Observable Markov Decision Processes (POMDP) provide strong theoretical foundations for action selection in the presence of uncertainty. However, exact solutions of POMDPs are only possible in few cases. Future work will investigate which approximations in the POMDP framework are necessary in the context of human-robot navigation.

Benchmarking. In the shared control chapter we have discussed three algorithms for navigation assistance. Future work will also investigate new frameworks to decision making under uncertainty such as POMDPs. This raises the question of how to evaluate a particular shared control algorithm in a way that allows algorithm-independent and objective comparisons. In the future we will investigate how to develop a sound framework for comparing different navigation assistance algorithms.

Experiments with real users. The final goal of human-robot navigation is to provide assistance to users who would otherwise be incapable of driving on their own. Therefore, the single most important direction of future work is to evaluate this architecture on a wide group of users and to prove that assistance effectively enhances user safety and driving comfort.

References

Aigner, P. and McCarragher, B. J. (2000). Modeling and constraining human interactions in shared control utilizing a discrete event framework. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):369–379.

Albrecht, D. W., Zukerman, I., and Nicholson, A. E. (1998). Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8:5–47.

Alexandersson, J. (1995). Plan recognition in verbmobil. *IJCAI Workshop The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI*, pages 2–7.

Allen, J. F. and Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.

Alt, H. and Godau, M. (1995). Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(1):75–91.

Alt, H., Knauer, C., and Wenk, C. (2003). Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58.

Arshak, K., Buckley, D., and Kaneswaran, K. (2006). Review of Assistive Devices for Electric Powered Wheelchairs Navigation. *ITB Journal*, 13:13–23.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *Robotics & Automation Magazine, IEEE*, 13(3):108–117.

Bauer, M. (1994). Quantitative modeling of user preferences for plan recognition. *Proceedings of the Fourth International Conference on User Modeling*, pages 73–78.

Bauer, M. (1996). A Dempster-Shafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction*, 5:317–348.

Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31.

Bennewitz, M., Burgard, W., and Thrun, S. (2002). Using EM to learn motion behaviors of persons with mobile robots. *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Borenstein, J. and Koren, Y. (1991). The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288.

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. *Proceedings of IEEE Intl. Conference on Robotics and Automation (ICRA)*, pages 341–346.

Brookmeyer, R., Gray, S., and Kawas, C. (1998). Projections of Alzheimer's disease in the United States and the public health impact of delaying disease onset. *American Journal of Public Health*, 88(9):1337–1342.

Buehler, M., Iagnemma, K., and Singh, S. (2007). *The 2005 DARPA Grand Challenge: The great robot race*. Springer Verlag.

Buehler, M., Iagnemma, K., and Singh, S. (2009). *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Verlag.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55.

Carberry, S. (1990). Incorporating default inferences into plan recognition. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 471–478.

Carberry, S. (2001). Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48.

Charniak, E. and Goldman, R. P. (1991). A probabilistic model of plan recognition. *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 160–165.

Charniak, E. and Goldman, R. P. (1993). A Bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79.

Compston, A. and Coles, A. (2002). Multiple Sclerosis. *Lancet*, 359:1221–31.

Cox, I. J. (1989). Blanche: Position estimation for an autonomous robot vehicle. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 432–439, Tsukuba, Japan.

Csató, L. (2008). Sparse Online Gaussian Processes (OGP) for Matlab. `http://www.kyb.tue.mpg.de/bs/people/csatol/ogp`.

Csato, L. and Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, 14:641–668.

Curtis (2010). Curtis instruments. http://www.curtisinstruments.com/.

De Schutter, J. (2010). Invariant description of rigid body motion trajectories. *Journal of Mechanisms and Robotics*, 2.

Demeester, E. (2007). *User-Adapted Plan Recognition And Shared Control for Wheelchair Driver Assistance under Uncertainty*. PhD thesis, Katholieke Universiteit Leuven.

Demeester, E., Hüntemann, A., Millán, J., and Van Brussel, H. (2009a). Augmenting Information from Brain-Computer Interfaces through Bayesian Plan Recognition. *Proceedings of European Symposium on Artificial Neural Networks (ESANN) - Advances in Computational Intelligence and Learning*, pages 479–489.

Demeester, E., Huntemann, A., and Van Brussel, H. (2009b). Bayesian plan recognition for brain-computer interfaces. *IEEE International Conference on Robotics and Automation, 2009. ICRA'09.*, pages 653–658.

Demeester, E., Huntemann, A., Vanhooydonck, D., Vanacker, G., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Bayesian estimation of wheelchair driver intents: Modeling intents as geometric paths tracked by the driver. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5775–5780.

Demeester, E., Huntemann, A., Vanhooydonck, D., Vanacker, G., Van Brussel, H., and Nuttin, M. (2008). User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Journal of Autonomous Robots*, 24(2):193–211.

Demeester, E., Nuttin, M., Vanhooydonck, D., and Van Brussel, H. (2003). A model-based, probabilistic framework for plan recognition in shared wheelchair control: Experiments and evaluation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1456–1461.

Demeester, E., Nuttin, M., Vanhooydonck, D., Vanacker, G., and Van Brussel, H. (2005). Global dynamic window approach for holonomic and non-holonomic mobile robots with arbitrary cross-section. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2694–2699.

Dempster, A., Laird, N., Rubin, D., et al. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M. (2001). A solution to the simultaneous localization and map building problem. *IEEE Transaction on Robotics and Automation*, 17(3):229–241.

Donnan, G. A., Fisher, M., and Davis, S. M. (2008). Stroke. *Lancet*, 371(9624):1612–1623.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13(2):99–110.

Dynamic (2010). Dynamic controls. http://www.dynamiccontrols.com/.

Edelmeyer, G., Van Brussel, H., Demeester, E., Hüntemann, A., Kronreif, G., Mayer, P., Vanacker, G., Vanhooydonck, D., Zagler, W. L., and Nuttin, M. (2005). Enhanced User Interface for Power Wheelchairs Developed in the MOVEMENT Project. *3rd European Medical & Biological Engineering Conference (EMBEC)*.

Fehr, L., Langbein, W. E., and Skaar, S. B. (2000). Adequacy of power wheelchair control interfaces for persons with severe disabilities: A clinical survey. *Journal of Rehabilitation Research and Development*, 37(3).

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, pages 23–33.

Fox, D., Burgard, W., Thrun, S., and Cremers, A. B. (1998). A hybrid collision avoidance method for mobile robots. *Proceedings of IEEE Intl. Conference on Robotics and Automation (ICRA)*, pages 1238–1243.

Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207.

Galán, F., Nuttin, M., Lew, E., Ferrez, P., Vanacker, G., Philips, J., and Millán, J. (2008). A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169.

Gertner, A. (1997). Plan recognition and evaluation for on-line critiquing. *User Modeling and User-Adapted Interaction*, 7(2):107–140.

Glover, J., Thrun, S., and Matthews, J. (2004). Learning user models of mobility-related activities through instrumented walking aids. 4:3306–3312.

Goedert, M. and Spillantini, M. G. (2006). A century of Alzheimer's Disease. *Science*, 314:777–781.

Goodman, B. A. and Litman, D. J. (1992). On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction*, 2:83–115.

Green, M. (2000). "How Long Does it Take to Stop?" Methodological analysis of driver perception-brake times. *Transportation Human Factors*, 2(3):195–216.

Guizzo, E. and Mandelbaum, R. (2010). DARPA seeking to revolutionize robotic manipulation. `http://spectrum.ieee.org/automaton/robotics/robotics-software/darpa-arm-program`.

Gutmann, J. S., Burgard, W., Fox, D., and Konolige, K. (1998). An experimental comparison of localization methods. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Heckerman, D. (1998). *Learning in graphical models*, chapter A tutorial on learning with Bayesian networks, pages 301–354. Kluwer Academic Publishers.

Heckerman, D., Geiger, D., and Chickering, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243.

Hess, R. A. and Modjtahedzadeh, A. (1990). A control theoretic model of driver steering behavior. *IEEE Control Systems Magazine*, 10(5):3–8.

HMC International (2010). Wheelchair interfaces. `http://www.hmc-products.com/`.

Hüntemann, A. (2005). Deliverable 3.1. framework for shared control and autonomy. Technical report, EU-IST MOVEMENT project, European Commision.

Hüntemann, A. (2006). Deliverable 3.2. first prototype implementation for self-adapting user interface. Technical report, EU-IST MOVEMENT project, European Commision.

Hüntemann, A. (2008). Deliverable 3.3. self-adapting user interface including user intention estimation. Technical report, EU-IST MOVEMENT project, European Commision.

Huntemann, A., Demeester, E., Nuttin, M., and Van Brussel, H. (2008a). Online user modelling with gaussian processes for bayesian plan recognition during power-wheelchair steering. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 285–292.

Huntemann, A., Demeester, E., Van Brussel, H., and Nuttin, M. (2008b). Can Bayes help disabled users? A Bayesian approach to plan recognition and shared control for steering an electrical wheelchair. Technical Report 470, Workshop at ACM/IEEE Conference on Human Robot Interaction (HRI) University of Hertfordshire, UK.

Huntemann, A., Demeester, E., Vanacker, G., Vanhooydonck, D., Philips, J., Van Brussel, H., and Nuttin, M. (2007a). Bayesian plan recognition and shared control under uncertainty: Assisting wheelchair drivers by tracking fine motion paths. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3360–3366.

Hüntemann, A., Demeester, E., Vander Poorten, E., and De Schutter, J. (2009). RADHAR: Robotic ADaptation to Humans Adapting to Robots: STREP project proposal Grant agreement no.: 248873. Technical report, European Commission.

Hüntemann, A., Demeester, E., Vanhooydonck, D., Vanacker, G., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Bayesian Estimation of Human Intent for Driving a Powered Wheelchair. *3rd International Workshop on Advances in Service Robotics (ASER)*.

Hüntemann, A., González, J., and Tapia, E. (2005). Tumor Classification from Gene Expression Data: A Coding-Based Multiclass Learning Approach. *International Symposium Biological and medical data analysis (ISBMDA). in Biological and Medical Data Analysis*, pages 211–222.

Huntemann, A., Mayer, P., Gelderblom, G., Pisetta, A., Kronreif, G., Einramhof, P., and Nuttin, M. (2007b). MOVEMENT use in progress. *IEEE 10th International Conference on Rehabilitation Robotics, (ICORR)*, pages 1096–1101.

Husqvarna (2011). Robotic lawn mowers. http://www.husqvarna.com/us/homeowner/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners/.

Huttenlocher, D., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, pages 850–863.

iRobot (2011). Roomba vacuum cleaner. http://www.irobot.com/.

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer, second edition.

Kantor, D. (2007). Medical encyclopedia. http://www.umm.edu/ency/index/eng_index.htm.

Katevas, N., Sgouros, N., Tzafestas, S., Papakonstantinou, G., Beattie, P., Bishop, J., Tsanakas, P., and Koutsouris, D. (1997). The autonomous mobile robot senario: A sensor-aided intelligent navigation system for powered wheelchairs. *IEEE Robotics and Automation*, 4(4):60–70.

Kautz, H. and Allen, J. (1986). Generalized plan recognition. *Proceedings of the Fifth National Conference on Artificial Intelligence.*, pages 32–37.

Kim, Y., Valtorta, M., and Vomlel, J. (2004). A prototypical system for soft evidential update. *Applied Intelligence*, 21(1):81–97.

Kleinman, D. L., Baron, S., and Levison, L. H. (1971). A control theoretic approach to manned-vehicle system analysis. *IEEE Transactions on Automatic Control*, 16(6):824–832.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Krigger, K. W. (2006). Cerebral Palsy: An overview. *American Family Physician*, 73(1):91–100.

Kronreif, G., Panek, P., Hüntemann, A., Cremers, G., Hochgatterer, A., Fürst, M., Mayer, P., and Gelderblom, G. (2008). MOVEMENT–A Modular and Versatile Mobility Enhancement System. *Computers Helping People with Special Needs*, pages 1244–1249.

Kronreif, G., Panek, P., Nuttin, M., Cremers, G., Pisetta, A., Fürst, M., Edelmeyer, G., and Hüntemann, A. (2007). *Conference for the Advancement of Assistive Technology in Europe (AAATE): in Challenges for assistive technology*, volume 20, pages 520–524. IOS Press.

LaValle, S. (2006). *Planning algorithms*. Cambridge University Press.

Lesh, N. (1997). Adaptive goal recognition. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1208–1214.

Lesh, N., Rich, C., and Sidner, C. L. (1999). Using plan recognition in human-computer collaboration. *Proceedings of the seventh international conference on User modeling*, pages 23–32.

Lin, Q. and Burdick, J. (2000). Objective and frame-invariant kinematic metric functions for rigid bodies. *The International Journal of Robotics Research*, 19(6):612–625.

Maas, A., Stocchetti, N., and Bullock, R. (2008). Moderate and severe traumatic brain injury in adults. *The Lancet Neurology*, 7(8):728–741.

Machado, A. (1912). *Proverbios y Cantares. XXIX. (Campos de Castilla).*

Marías, J. (2007). *Your face tomorrow: Fever and spear*, volume 1. New Directions Publishing Corporation.

Marías, J. (2008). *Your face tomorrow: Dance and dream*, volume 2. New Directions Publishing Corporation.

Martinez, J. and Duffy, J. (1995). On the metrics of rigid body displacements for infinite and finite bodies. *Journal of Mechanical Design*, 117:41.

Mayer, P., Edelmayer, G., Gelderblom, G. J., Vincze, M., Einramhof, P., Nuttin, M., Fuxreiter, T., and Kronreif, G. (2007). Movement –modular versatile mobility enhancement system. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2892 – 2897.

Mayfield, J. (1992). Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, 2:55–82.

McDonald, J. W. and Sadowsky, C. (2002). Spinal-cord injury. *Lancet*, 359(9304):417–425.

McRuer, D. T., Graham, D., Krendel, E. S., and W. Reisener, J. (1965). Human pilot dynamics in compensatory systems - theory, models, and experiments with controlled element and forcing function variations. Technical report, AFFDL-TR-65-15 U.S. Air Force.

McRuer, D. T. and Jex, H. R. (1967). A review of quasi-linear pilot models. *IEEE Transactions on Human Factors in Electronics*, 8(3):231–240.

McRuer, D. T. and Krendel, E. S. (1957). Dynamic response of human operators. Technical report, WADC-TR-56-524 U.S. Air Force.

McRuer, D. T. and Weir, D. (1969). Theory of manual vehicular control. *Ergonomics*, 12(4):599–633.

Millnert, O., Goedemé, T., Tuytelaars, T., Van Gool, L., Hüntemann, A., and Nuttin, M. (2007). Range determination for mobile robots using an omnidirectional camera. *Integrated Computer-Aided Engineering*, 14(1):63–72.

Mittal, V., Yanco, H., Aronis, J., and Simpson, R. (1998). *Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces and Natural Language Processing.*

Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA)*, pages 1985–1991.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the National conference on Artificial Intelligence*, pages 593–598.

Mulisch, H. (1998). *The discovery of heaven*. Penguin.

Murphy, K. (2002). *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley.

Murphy, K. (2010). Bayes Net Toolbox for Matlab. `http://code.google.com/p/bnt/`.

Newton, J. (1989). *Uncommon Friends: Life with Thomas Edison, Henry Ford, Harvey Firestone, Alexis Carrel, and Charles Lindbergh*. Mariner Books.

Olanow, C. W. and Tatton, W. G. (1999). Etiology and pathogenesis of Parkinson's Disease. *Annual Review of Neuroscience*, 22(1):123–144.

Opper, M. (1998). *A Bayesian Approach to Online Learning*, pages 363–378. On-line Learning in Neural Networks. Cambridge University Press.

Papoulis, A. (1991). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Companies, 3rd edition.

Parikh, S. P., Jr., V. G., Kumar, V., and Jr., J. O. (2004). Incorporating user inputs in motion planning for a smart wheelchair. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2048.

Park, F. (1995). Distance metrics on the rigid-body motions with applications to mechanism design. *Journal of mechanical design*, 117(1):48–54.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Permobil (2010). Wheelchair interfaces. `http://www.permobil.com`.

Prassler, E., Scholz, J., and Fiorini, P. (1999). Navigating a robotic wheelchair in a railway station during rush hour. *The International Journal of Robotics Research*, 18(7):711–727.

RADHAR (2010). Robotic ADaptation to Humans Adapting to Robots. `http://www.radhar.eu/`.

Rasmussen, C. E., Nickisch, H., and Williams, C. (2010). Gaussian Processes for Machine Learning (GPML) in Matlab. `http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html`.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, second edition.

Retz-Schmidt, G. (1991). Recognizing intentions, interactions, and causes of plan failures. *User Modeling and User-Adapted Interaction*, 1:173–202.

Robomow (2011). Robotic lawn mowers. `http://www.robomow.com/`.

Röfer, T. and Lankenau, A. (2000). Architecture and applications of the bremen autonomous wheelchair. *Information Sciences*, 126(1):1–20.

Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. (1978). The plan recognition problem. *Artificial Intelligence*, 11:45–83.

Setz, C., Arnrich, B., Schumm, J., La Marca, R., Troster, G., and Ehlert, U. (2010). Discriminating stress from cognitive load using a wearable EDA device. *Information Technology in Biomedicine, IEEE Transactions on*, 14(2):410–417.

Sheridan, T. B. and Ferrel, W. R. (1981). *Information, Control and Decision Models of Human Performance*. The MIT Press.

Shonkwiler, R. (1991). Computing the Hausdorff set distance in linear time for any L_p point distance. *Information Processing Letters*, 38(4):201–207.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3375–3382.

Simpson, R. C. and Levine, S. P. (1999). Automatic adaption in the navchair assistive wheelchair navigation system. *IEEE Transactions on Rehabilitation Engineering*, 7(4):452–463.

Simpson, R. C., Levine, S. P., Bell, D. A., Jaros, L. A., Koren, Y., and Borenstein, J. (1998). Navchair: An assistive wheelchair navigation system with automatic adaptation. *Lecture Notes in Artificial Intelligence*, 1458:235–255.

Simpson, R. C., LoPresti, E. F., and Cooper, R. A. (2008). How many people would benefit from a smart wheelchair? *Journal of Rehabilitation Research & Development*, 45(1):53–72.

Special Issue RAM (2001). Reinventing the wheelchair - autonomous robotic wheelchair projects in Europe improve mobility and safety. *IEEE Robotics and Automation Magazine*, 8(1).

Therafin (2010). Sip and puff wheelchair interface. `http://www.therafin.com/sipnpuff.htm`.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press, first edition.

Valtorta, M., Kim, Y., and Vomlel, J. (2002). Soft evidential update for probabilistic multiagent systems. *International Journal of Approximate Reasoning*, 29(1):71–106.

Vanacker, G. (2008). *Adaptive steering behaviour modelling for power wheelchair control*. PhD thesis, Katholieke Universiteit Leuven.

Vanacker, G., del R. Millán, J., Lew, E., Ferrez, P. W., Moles, F. G., Philips, J., Van Brussel, H., and Nuttin, M. (2007). Context-based filtering for assisted brain-actuated wheelchair driving. *Computational Intelligence and Neuroscience*, 2007:3.

Vanacker, G., Vanhooydonck, D., Demeester, E., Huntemann, A., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Adaptive filtering approach to improve wheelchair driving performance. *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN*, pages 527–532.

Vanhooydonck, D. (2007). *Shared Control for Intelligent Wheelchairs: Estimation of the User Intention based on an Implicit Personalized User Model*. PhD thesis, Katholieke Universiteit Leuven.

Vanhooydonck, D., Demeester, E., Hüntemann, A., Philips, J., Vanacker, G., Van Brussel, H., and Nuttin, M. (2010). Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robotics and Autonomous Systems*, 58(8):963–977.

Vecchio, A., Huentemann, A., Van der Auweraer, H., Verhaeven, E., and Lenaers, G. (2005a). NVH Testing on Modern Vehicle Power Train Technologies. *Proceedings of the 12th International Congress on Sound and Vibration (ICSV)*, 12.

Vecchio, A., Huntemann, A., Van der Auweraer, H., and Leuridan, J. (2005b). Multi-agent Systems in a One-man Operation Measurement Setup. *IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, (IDAACS)*, pages 95–100.

Vecchio, A., Huntemann, A., Van der Auweraer, H., and Leuridan, J. (2006). Use of Interface Agents for Automatic Pass-by-Noise Measurements. *Proceedings of the IEEE Instrumentation and Measurement Technology Conference, (IMTC)*, pages 16–21.

Walling, A. D. (1999). Amyotrophic Lateral Sclerosis: Lou Gehrig's disease. *American Family Physician*, 59(6):1489–1496.

Weir, D. and McRuer, D. T. (1970). Dynamics of driver vehicle steering control. *Automatica*, 6:87–98.

Wenninga, G. K., Colosimo, C., Gesera, F., and Poewea, W. (2004). Multiple System Atrophy. *The Lancet Neurology*, 3(2):93–103.

Wiki Commons: C. Rocchini (2011). Example hausdorff distance. `http://en.wikipedia.org/wiki/User:Rocchini`.

Wiki Commons: Q. Jarosz (2010). Schema of a neuron. `http://en.wikipedia.org/wiki/User:Quasar_Jarosz`.

Yanco, H. (2000). *Shared User-Computer Control of a Robotic Wheelchair System*. PhD thesis, Massachusetts Institute of Technology.

Yanco, H. A. (1998). Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. *Lecture notes in artificial intelligence*, pages 256–268.

Young, L. R. (1969). On adaptive manual control. *IEEE Transactions on Man Machine Systems*, 10(4):292–331.

Zampieri, C. and Fabio, R. P. D. (2006). Progressive supranuclear palsy: Disease profile and rehabilitation strategies. *Physical Therapy*, 86(6):870–880.

Curriculum Vitae

Personal details

Name: Alexander Hüntemann
Date and place of birth: 22/02/1980 in Madrid (Spain)
Nationality: German

Education

2005-2011 Ph.D. at the Katholieke Universiteit Leuven,
Mechanical Engineering Department

1998 - 2003 Escuela Técnica Superior de Ingenieros de Telecomunicación,
Universidad Politénica de Madrid (Technical University Madrid)

December 2003 Master in Telecommunications Engineering
(Ranking: 7th out of 404 in the class of 2003).

1986 - 1998 German Secondary School of Madrid

May 1998 1.0 German Abitur (German university access exam)

June 1998 Sobresaliente in Spanish Selectividad (Spanish university access exam)

Professional experience

2010-present Member coordination team of EU RADHAR project

2005-2009 Assistant of the Research Foundation Flanders (FWO)

2005-2008 EU MOVEMENT project

2004 Marie Curie Fellow at the company LMS International (Belgium)

List of publications

Articles in internationally reviewed journals

1. Vanhooydonck, D., Demeester, E., Hüntemann, A., Philips, J., Vanacker, G., Van Brussel, H., and Nuttin, M. (2010). Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robotics and Autonomous Systems*, 58(8):963–977

2. Demeester, E., Huntemann, A., Vanhooydonck, D., Vanacker, G., Van Brussel, H., and Nuttin, M. (2008). User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Journal of Autonomous Robots*, 24(2):193–211

3. Millnert, O., Goedemé, T., Tuytelaars, T., Van Gool, L., Hüntemann, A., and Nuttin, M. (2007). Range determination for mobile robots using an omnidirectional camera. *Integrated Computer-Aided Engineering*, 14(1):63–72

Papers at international peer-reviewed conferences

1. Demeester, E., Huntemann, A., and Van Brussel, H. (2009b). Bayesian plan recognition for brain-computer interfaces. *IEEE International Conference on Robotics and Automation, 2009. ICRA'09.*, pages 653–658

2. Demeester, E., Hüntemann, A., Millán, J., and Van Brussel, H. (2009a). Augmenting Information from Brain-Computer Interfaces through Bayesian Plan Recognition. *Proceedings of European Symposium on Artificial Neural Networks (ESANN) - Advances in Computational Intelligence and Learning*, pages 479–489

3. Huntemann, A., Demeester, E., Nuttin, M., and Van Brussel, H. (2008a). Online user modelling with gaussian processes for bayesian plan recognition during power-wheelchair steering. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 285–292

4. Kronreif, G., Panek, P., Hüntemann, A., Cremers, G., Hochgatterer, A., Fürst, M., Mayer, P., and Gelderblom, G. (2008). MOVEMENT–A Modular and Versatile Mobility Enhancement System. *Computers Helping People with Special Needs*, pages 1244–1249

5. Huntemann, A., Demeester, E., Vanacker, G., Vanhooydonck, D., Philips, J., Van Brussel, H., and Nuttin, M. (2007a). Bayesian plan recognition and shared control under uncertainty: Assisting wheelchair drivers by tracking fine motion paths. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3360–3366

6. Huntemann, A., Mayer, P., Gelderblom, G., Pisetta, A., Kronreif, G., Einramhof, P., and Nuttin, M. (2007b). MOVEMENT use in progress. *IEEE 10th International Conference on Rehabilitation Robotics, (ICORR)*, pages 1096–1101

7. Kronreif, G., Panek, P., Nuttin, M., Cremers, G., Pisetta, A., Fürst, M., Edelmeyer, G., and Hüntemann, A. (2007). *Conference for the Advancement of Assistive Technology in Europe (AAATE): in Challenges for assistive technology*, volume 20, pages 520–524. IOS Press

8. Demeester, E., Huntemann, A., Vanhooydonck, D., Vanacker, G., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Bayesian estimation of wheelchair driver intents: Modeling intents as geometric paths tracked by the driver. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5775–5780

9. Vanacker, G., Vanhooydonck, D., Demeester, E., Huntemann, A., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Adaptive filtering approach to improve wheelchair driving performance. *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN*, pages 527–532

10. Vecchio, A., Huntemann, A., Van der Auweraer, H., and Leuridan, J. (2006). Use of Interface Agents for Automatic Pass-by-Noise Measurements. *Proceedings of the IEEE Instrumentation and Measurement Technology Conference, (IMTC)*, pages 16–21

11. Vecchio, A., Huntemann, A., Van der Auweraer, H., and Leuridan, J. (2005b). Multi-agent Systems in a One-man Operation Measurement Setup. *IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, (IDAACS)*, pages 95–100

12. Hüntemann, A., González, J., and Tapia, E. (2005). Tumor Classification from Gene Expression Data: A Coding-Based Multiclass Learning Approach. *International Symposium Biological and medical data analysis (ISBMDA). in Biological and Medical Data Analysis*, pages 211–222

13. Vecchio, A., Huentemann, A., Van der Auweraer, H., Verhaeven, E., and Lenaers, G. (2005a). NVH Testing on Modern Vehicle Power Train Technologies. *Proceedings of the 12th International Congress on Sound and Vibration (ICSV)*, 12

Workshops

1. Huntemann, A., Demeester, E., Van Brussel, H., and Nuttin, M. (2008b). Can Bayes help disabled users? A Bayesian approach to plan recognition and shared control for steering an electrical wheelchair. Technical Report 470, Workshop at ACM/IEEE Conference on Human Robot Interaction (HRI) University of Hertfordshire, UK

2. Hüntemann, A., Demeester, E., Vanhooydonck, D., Vanacker, G., Degeest, A., Van Brussel, H., and Nuttin, M. (2006). Bayesian Estimation of Human Intent for Driving a Powered Wheelchair. *3rd International Workshop on Advances in Service Robotics (ASER)*

3. Edelmeyer, G., Van Brussel, H., Demeester, E., Hüntemann, A., Kronreif, G., Mayer, P., Vanacker, G., Vanhooydonck, D., Zagler, W. L., and Nuttin, M. (2005). Enhanced User Interface for Power Wheelchairs Developed in the MOVEMENT Project. *3rd European Medical & Biological Engineering Conference (EMBEC)*

Technical reports

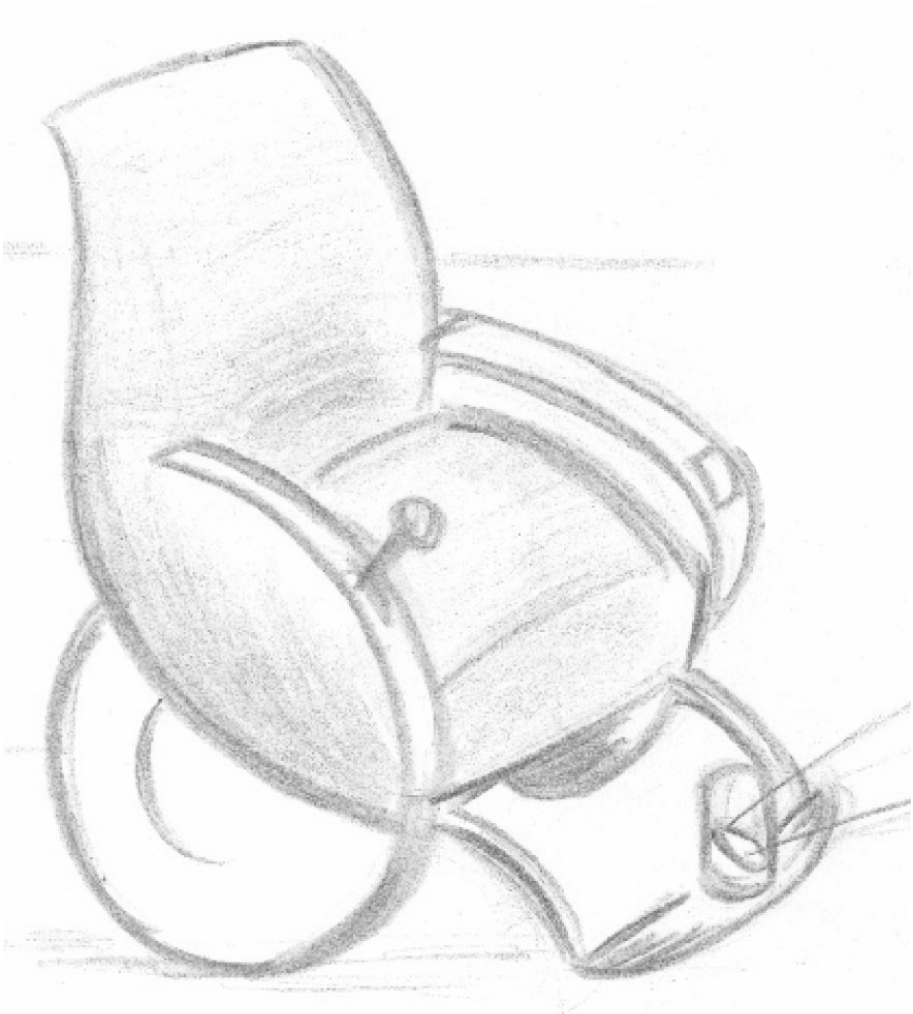
1. Hüntemann, A., Demeester, E., Vander Poorten, E., and De Schutter, J. (2009). RADHAR: Robotic ADaptation to Humans Adapting to Robots: STREP project proposal Grant agreement no.: 248873. Technical report, European Commission

2. Hüntemann, A. (2008). Deliverable 3.3. self-adapting user interface including user intention estimation. Technical report, EU-IST MOVEMENT project, European Commision

3. Hüntemann, A. (2006). Deliverable 3.2. first prototype implementation for self-adapting user interface. Technical report, EU-IST MOVEMENT project, European Commision

4. Hüntemann, A. (2005). Deliverable 3.1. framework for shared control and autonomy. Technical report, EU-IST MOVEMENT project, European Commision

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Mechanical Engineering
Production engineering, Machine design and Automation (PMA)
Celestijnenlaan 300B, B-3001 Leuven (Belgium)



KATHOLIEKE UNIVERSITEIT LEUVEN
ASSOCIATIE K.U.LEUVEN