© Katholieke Universiteit Leuven
Faculteit Toegepaste Wetenschappen
Arenbergkasteel, B-3001 Heverlee (Leuven), Belgium

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

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

D/2005/7515/71
ISBN 90-5682-637-9
UDC 681.3∗I29

Voorwoord

"You get a lot of scientists, particularly American scientists, saying that robotics is about at the level of the rat at the moment, I would say it's not anywhere near even a simple bacteria."

Prof. Noel Sharkey, Sheffield University, 090305

Ik was reeds begonnen met een filosofische behandeling van bovenstaande citatie. Niet dus. Schrijven is schrappen[1] en er volgt nog genoeg over robotica in het vervolg van dit boek(je). Ik laat het nadenken over bovenstaande uitspraak dus volledig aan u over.

U leest dit voorwoord immers waarschijnlijk om vast te stellen of ik u niet vergeten te bedanken ben, of omdat u alleen maar tijd en zin heeft om het voorwoord te lezen.[2]

Mogelijkheid één. Wie moet ik allemaal bedanken? Wie moet ik met naam en toenaam vermelden, en wie vergeet ik te vermelden? Een random number generator die mensen uit mijn adresboek selecteert op basis van het aantal verstuurde emails lijkt me niet de beste optie. Wie droeg rechtstreeks bij tot het doctoraat, en wie onrechtstreeks? Als u dit leest, is de kans heel groot dat ik u moet bedanken. Dank je wel! En ook bedankt aan diegenen dit nooit zullen lezen, maar toch bewust of onbewust aan dit proefschrift hebben meegewerkt.

Mogelijkheid twee. Alleen maar tijd en zin om het voorwoord te lezen. Daar heeft u waarschijnlijk ondertussen al spijt van. Mijn zinsconstructies

[1] Voor de nerds onder u, svn versie 5558 van de actsens repository bevat nog de oude versie.

[2] Bemerk dus beste lezer, dat we vertrekken van een discrete a priori dichtheid met twee mogelijkheden en wanneer we kansen moeten toekennen aan elk van die twee mogelijkheden, kan de *believer–vs–non-believer* discussie over Bayesiaanse waarschijnlijkheidsleer losbarsten. Tussen haakjes: voetnoten in het voorwoord moet je mijden als de pest!

en gedachten lijken immers maar al te vaak op een programma-stack[3] en mijn instructiepointer verspringt vaak naar een ander gedeelte van mijn uitleg vooraleer ik erin slaag om de volledige context beschreven te hebben. Wat vaak resulteert in lange zinnen, met veel bijzinnen, haakjesconstructies en voetnoten, wat het geheel moeilijk leesbaar maakt. Niet?

Ziezo. Ik vraag me af of ik ooit nog een voorwoord zal moeten schrijven.

"If you do the job badly enough, sometimes you don't get asked to do it again."

Bill Watterson in "Homicidal Psycho Jungle Cat"

ps. De rest van de tekst is (gelukkig) iets serieuzer...

Klaas.
Leuven, 23 augustus 2005.

[3]Gelukkig voor u lijken ze nog op de *stack* en niet op de *heap*.

Abstract

Autonomous compliant motion (ACM) is one important skill in view of deploying robots that have to interact physically with an unstructured environment. Typical applications are assembly tasks where precise positioning of the objects is impossible (e.g. in space or sub-sea applications), industrial robots used for machining workpieces without the need for costly and time-consuming fixturing and off-line measuring, and service robots that have to operate in human environments where the objects to be manipulated have no fixed location. This thesis focuses on one aspect of ACM: the estimation of unknown geometrical parameters, e.g. the location and/or dimension of objects in the robot's environment.

This work develops a Bayesian approach to deal with large uncertainties on the geometrical parameters due to uncertainty in, both, the location and the shape of the interacting objects, as well as due to uncertainty on which geometric primitives of robot and environment are in contact with each other. An explicit, hybrid model allows to accurately predict contact formation transitions based on information about the current value of geometrical parameters and contact formation, and a high level task plan. This model is an extension of the well known Hidden Markov Models with unknown parameters. Sequential Monte Carlo methods are used to perform online estimation in these models. The hybrid model can also be used as a generalization of Jump Markov Models, so its scope is broader than the applications dealt with in this thesis.

This thesis also presents a classification of estimation algorithms based on the nature of the Random Variables. This classification provides a much improved understanding of the relation between the choice of probabilistic models and the complexity of algorithmic implementations, and it helps robot practitioners without a deep background in estimation to choose a particular algorithm that corresponds best to the world and interaction models that they use.

The classification also provides the structure for the Bayesian Filtering Library (BFL) developed during this work. BFL is an open source soft-

ware library that does not impose restrictions on the nature of the Random Variables—both discrete, continuous and hybrid states/parameters are allowed—nor on the representation of the posterior PDF (analytical, sample based, ...). Its use is certainly not limited to autonomous compliant motion.

Beknopte Samenvatting

Het autonoom uitvoeren van robottaken in contact (Eng. Autonomous Compliant Motion, ACM) is één belangrijke vaardigheid die nodig is om robots in te zetten in ongestructureerde omgevingen waar ze fysisch mee moeten interageren. Typische toepassingen zijn assemblagetaken waar de preciese positionering van de voorwerpen onmogelijk is (zoals in toepassingen voor ruimtevaart of onder het wateroppervlak), industriële robots die werkstukken moeten bewerken zonder gebruik te maken van tijdrovende klemmingen en offline meettechnieken, en service robots die moeten opereren in omgevingen waar mensen leven en waar de objecten die ze moeten manipuleren geen vaste plaats hebben. Deze thesis concentreert zich op één aspect van ACM: het schatten van onbekende geometrische parameters zoals de locatie en/of dimensies van voorwerpen uit de robots omgeving.

Dit werk ontwikkelt een Bayesiaanse aanpak die toelaat om te gaan met grote onzekerheden op de geometrische parameters. De onzekerheden zijn te wijten aan de onbekende locatie en vorm van de objecten die met elkaar in contact zijn enerzijds, en de onzekerheid over *welke* geometrische primitieven van robot en omgeving met elkaar in contact zijn. Een expliciet hybride model laat toe om contactformatietransities te voorspellen gebaseerd op de huidige contactformatie, de waarde van de geometrische parameters en een hoog niveau taakplan. Dit model is een uitbreiding op Verborgen Markov Modellen met onbekende parameters. Recursieve Monte Carlo methodes schatten de geometrische parameters online aan de hand van bovenstaande modellen. Het hybride model kan ook gezien worden als een veralgemening van Jump Markov Modellen, wat inhoudt dat het toepassingsgebied ervan breder is dan de toepassingen uit deze thesis.

Deze thesis beschrijft ook een classificatie van estimatie-algoritmes gebaseerd op de aard van de kansvariabelen. Deze classificatie verhoogt de bestaande inzichten in de relaties tussen de keuze van probabilistische modellen en de complexiteit van algoritmische implementaties. Ze helpt roboticaspecialisten, die niet noodzakelijk specialisten zijn in estimatietechnieken, om het algoritme te kiezen dat best past bij de interactie– en wereldmodellen die ze

gebruiken.

De structuur van de classificatie is ook de basis voor de Bayesiaanse estimatiebibliotheek (BFL) ontwikkeld in deze thesis. BFL is een open bron softwarebibliotheek die geen restricties oplegt met betrekking tot de aard van de kansvariabelen—zowel discrete, continue als hybride toestands– en/of parametervariabelen zijn toegelaten—noch met betrekking tot de voorstelling van de a posteriori PDF (analytisch, Monte Carlo representatie, ...). Het toepassingsgebied van de bibliotheek is veel breder dan ACM.

Symbols, definitions and abbreviations

General abbreviations

1D, 2D, 3D	: 1-, 2-, or 3-dimensional
ADF	: Assumed Density Filter
ASIR	: Auxialiary Sequential Importance Sampling with Resampling
BFL	: Bayesian Filtering Library
BN	: Bayesian Network
CC	: Contact Configuration
CDF	: Cumulative Distribution Function
CF	: Contact Formation
CKF	: Cascaded Kalman Filter
CLT	: Central Limit Theorem
DAG	: Directed Acyclic Graph
DBN	: Dynamic Bayesian Network
EKF	: Extended Kalman Filter
EM	: Expectation–Maximization
EP	: Expectation–Propagation
ESS	: Effective Sample Size
GPB	: General Pseudo Bayes
HMM	: Hidden Markov Model
IEKF	: Iterated Extended Kalman Filter
IMM	: Interacting Multiple Models
JM(L)S	: Jump Markov (Linear) System
KF	: Kalman Filter
KLD	: Kullback-Leibler Distance
MAP	: Maximum A Posteriori
MC	: Monte Carlo
MCMC	: Markov Chain Monte Carlo
MDP	: Markov Decision Process
ML	: Maximum Likelihood

NIS	: Normalised Innovation Squared
NMSKF	: Non-minimal State Kalman Filter
NN	: Nearest Neighbour
PDF	: Probability Density Function
PF	: Particle Filter
POMDP	: Partially Observable Markov Decision Process
RNG	: Random Number Generator
RV	: Random Variable
SIS	: Sequential Importance Sampling
SIR/SISR	: Sequential Importance Sampling with Resampling stage
SNIS	: Summed Normalised Innovation Squared
TJTF	: Thin Junction Tree Filter

General symbols and definitions

a	: scalar (unbold lower case)
$\boldsymbol{a}$	: vector (bold lower case)
$\boldsymbol{A}$	: matrix (bold upper case)
$\boldsymbol{A}_{m\times n}$	: m-by-n matrix $\boldsymbol{A}$
$\boldsymbol{a}(i)$	: ith element of the vector $\boldsymbol{a}$
$\boldsymbol{a}^T$ or $\boldsymbol{A}^T$	: transpose of the vector $\boldsymbol{a}$ or the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	: inverse of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{\#}$	: generalised inverse of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{\#}_{W}$	: generalised inverse of the matrix $\boldsymbol{A}$ with weighting matrix $\boldsymbol{W}$
$\lvert\boldsymbol{A}\rvert$	: determinant of the matrix $\boldsymbol{A}$
$\mathrm{diag}\,(\sigma_1,\ldots,\sigma_n)$	: n-by-n diagonal matrix with diagonal elements $\sigma_1,\ldots,\sigma_n$
$\boldsymbol{a}\times\boldsymbol{b}$	: vector product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$[\boldsymbol{p}\times]$	: matrix representing the vector product with $\boldsymbol{p}$
$\boldsymbol{I}_{m\times m}$	: m-by-m identity matrix
$\boldsymbol{0}_{m\times n}$	: m-by-n zero matrix
$\boldsymbol{0}_{m\times 1}$	: m-dimensional zero vector
$\hat{a}$ or $\hat{\boldsymbol{a}}$	: value for the scalar a or the vector $\boldsymbol{a}$
$\cdot_k$	: discrete time index
$\cdot_{k\lvert k-1}$	: at time k given the measurements up to time $k-1$
$\cdot_{k\lvert k}$	: at time k given the measurements up to time k

Bayesian probability theory

A	: scalar random variable
$\boldsymbol{A}$	: vector valued random variable
$P(A=a)$ or $P(a)$	: probability of $A=a$
$P(A=a\lvert B=b)$	: probability of $A=a$ given $B=b$

$\boldsymbol{x}$	: n-dimensional state vector
$\boldsymbol{\theta}$	: n-dimensional parameter vector
$\boldsymbol{x}^*$	: true value of the state vector
$\hat{\boldsymbol{x}}$	: state estimate
$\hat{\boldsymbol{\Sigma}}$	: estimate of the covariance matrix of $P(A)$
$\boldsymbol{\Sigma}$	: covariance matrix of $P(A)$
$\boldsymbol{I}$	: information matrix of the state estimate
$\boldsymbol{z}$	: m-dimensional measurement vector
$\hat{\boldsymbol{z}}$	: measured value
$\boldsymbol{u}$	: system input
$\boldsymbol{s}$	: sensor input
$\boldsymbol{x}_{1:k}$	: sequence of state vectors from $t = 1$ to $t = k$
$\boldsymbol{z}_{1:k}$	: sequence of measurement vectors
$\boldsymbol{u}_{1:k}$	: sequence of system inputs
$\boldsymbol{s}_{1:k}$	: sequence of sensor inputs
$p(\boldsymbol{a})$	: prior PDF over the RV $\boldsymbol{a}$
$p(\boldsymbol{a}\|\boldsymbol{z})$	: posterior PDF over the RV $\boldsymbol{a}$ given the measurement $\boldsymbol{z}$
$p(\boldsymbol{a}\|\boldsymbol{z}_{1:k})$	: posterior PDF over the RV $\boldsymbol{a}$ given the measurements $\boldsymbol{z}_{1:k}$
$p(\boldsymbol{z}\|\boldsymbol{a})$	: likelihood of the measurement $\boldsymbol{z}$ as a function of the RV $\boldsymbol{a}$
$p(\boldsymbol{z}_{1:k}\|\boldsymbol{a})$	: likelihood of the measurements $\boldsymbol{z}_{1:k}$ as a function of the RV $\boldsymbol{a}$
$\boldsymbol{f}(.)$	: system (process) model
$\boldsymbol{h}(.)$	: measurement model
$E_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$	: Expected value of the function $h(\boldsymbol{x})$ given the distribution $p(\boldsymbol{x})$
$Var_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$	: Variance of the function $h(\boldsymbol{x})$ given the distribution $p(\boldsymbol{x})$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	: Gaussian PDF
$U[a, b]$	: Uniform PDF on the interval [a,b]
$D(.\|\|.)$	: Kullback-Leibler distance

Monte Carlo methods & Particle Filtering

$q(\cdot)$	: Proposal density
$w_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta})$	: Particle weights at timestep k
$\hat{w}_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta})$	: Incremental Particle weights at timestep k
$\tilde{w}_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta})$	: Normalized Particle weights at timestep k
$\pi(\boldsymbol{x}_k\|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k})$	: Optimal recursive proposal density at timestep k
a	: Acceptance ratio for MCMC algorithms
I	: true value of sum/integral, typically $E_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$
$\hat{I}$	: approximation of I
N	: number of samples
ESS	: effective sample size
u_i	: i^{th} uniform sample
$T(\cdot)$	: Markov Transition Kernel

Contact modelling

$\boldsymbol{t}$	: twist
$\boldsymbol{v}$	: translational velocity
$\boldsymbol{\omega}$	: rotational velocity
$\boldsymbol{w}$	: wrench
$\boldsymbol{f}$	: force
$\boldsymbol{m}$	: moment
$\boldsymbol{t}_d$	: pose
$\boldsymbol{G}$	: wrench spanning set
$\boldsymbol{\phi}$	: wrench decomposition
$\boldsymbol{J}$	: twist spanning set
$\boldsymbol{\chi}$	: twist decomposition
$\boldsymbol{h}_d(.) = \boldsymbol{0}$	: closure equation
{w}	: world frame
{e}	: environment frame
{i}	: environment frame for the ith environment object
{g}	: gripper frame
{m}	: manipulated object frame
{c}	: frame in a contact point
$\boldsymbol{\theta}^m$	: grasping uncertainties $\boldsymbol{\theta}^m = [x^m\ y^m\ z^m\ \theta_x^m\ \theta_y^m\ \theta_z^m]^T$
x^m, y^m, z^m	: position of the manipulated object
$\theta_x^m, \theta_y^m, \theta_z^m$	: orientation of the manipulated object
$\boldsymbol{\theta}^{dm}$	: uncertain dimensions of the manipulated object
$\boldsymbol{\theta}^e$	: environment uncertainties $\boldsymbol{x}^e = [x^e\ y^e\ z^e\ \theta_x^e\ \theta_y^e\ \theta_z^e]^T$
x^e, y^e, z^e	: position of the environment object
$\theta_x^e, \theta_y^e, \theta_z^e$	: orientation of the environment object
$\boldsymbol{\theta}^{de}$	: uncertain dimensions of the environment object
$\boldsymbol{\theta}^i$	: environment uncertainties for the ith environment object $\boldsymbol{\theta}^i = [x^i\ y^i\ z^i\ \theta_x^i\ \theta_y^i\ \theta_z^i]^T$
x^i, y^i, z^i	: position of the ith environment object
$\theta_x^i, \theta_y^i, \theta_z^i$	: orientation of the ith environment object
$\boldsymbol{\theta}^{di}$	: uncertain dimensions of the ith environment object
${}^a_b\boldsymbol{S}_w$	: wrench transformation matrix to transform a wrench expressed in {a} to a wrench expressed in {b}
${}^a_b\boldsymbol{S}_t$	: twist transformation matrix to transform a twist expressed in {a} to a twist expressed in {b}
${}^a_b\boldsymbol{R}$	: rotation matrix to rotate a vector from {a} to {b}
${}_c\boldsymbol{p}^{b,a}$	: vector from the origin of {b} to the origin of {a} expressed in the frame {c}

${}^{a}_{b}\boldsymbol{T}$	: homogeneous transformation matrix to transform a point expressed in {a} to a point expressed in {b}
$\boldsymbol{H}, \boldsymbol{d}, \boldsymbol{E}, \boldsymbol{D}$	: linearization of an implicit, nonlinear measurement model

Table of contents

Voorwoord i

Abstract iii

Beknopte Samenvatting v

Symbols, definitions and abbreviations vii

Table of contents xiii

1 Introduction **1**
1.1 Introduction 1
1.2 A system for ACM 4
1.3 Applications of this research 6
1.4 The Bayesian approach 8
1.5 Contributions 9
1.6 Overview of this thesis 10

2 Bayesian Inference **13**
2.1 Introduction 13
2.2 The joint posterior 14
2.3 Applications 15
2.4 Filtering 16
2.5 Bayes' rule 17
2.5.1 ML and MAP 19
2.5.2 Conjugacy 21
2.6 (Graphical) Modeling—Bayesian Networks 21
2.7 Decision making 25
2.8 Conclusions 27

3 Monte Carlo methods **29**
3.1 Introduction . . . 29
3.2 Monte Carlo methods . . . 31
3.2.1 Convergence of Monte Carlo methods . . . 32
3.2.2 Performance of Monte Carlo methods . . . 33
3.3 Sampling from a discrete distribution . . . 34
3.4 Inversion sampling . . . 34
3.5 Importance sampling . . . 37
3.5.1 Obtaining approximate samples from a PDF . . . 37
3.5.2 Monte Carlo Integration using Importance Sampling 39
3.5.3 Performance of Importance Sampling methods . . . 39
3.5.4 Choosing the proposal density . . . 41
3.5.5 Applications . . . 41
3.6 Rejection sampling . . . 42
3.7 Markov Chain Monte Carlo (MCMC) methods . . . 43
3.7.1 The Metropolis–Hastings algorithm . . . 44
3.7.2 MCMC variants . . . 47
3.7.3 Conclusions . . . 48
3.8 Overview and Conclusions . . . 48

4 Sequential Monte Carlo methods **51**
4.1 Introduction . . . 51
4.2 The Sequential Importance Sampling algorithm . . . 52
4.3 Choosing the proposal density . . . 54
4.3.1 The optimal proposal density . . . 55
4.3.2 Using an analytic filter in the proposal step . . . 56
4.3.3 Other proposal densities . . . 56
4.4 Impoverishment . . . 58
4.4.1 Resampling . . . 59
4.4.2 Application of a MCMC step . . . 61
4.4.3 Methods based on rejection control . . . 63
4.5 Convergence of Particle Filters . . . 63
4.6 Choosing the number of particles . . . 64
4.7 Software and hardware implementations . . . 65
4.8 Conclusions . . . 66

5 Classification of Bayesian inference algorithms **67**
5.1 Introduction . . . 67
5.2 Classification of the joint posterior . . . 68
5.3 Pure State Estimation, known parameters . . . 71
5.3.1 Continuous states, known parameters . . . 71
5.3.2 Discrete states, known parameters . . . 73
5.3.3 Hybrid states, known parameters . . . 74

5.3.4 Data association 79
5.4 Pure Parameter Estimation 85
5.4.1 Known states, continuous parameters 85
5.4.2 Known states, discrete parameters 88
5.4.3 Known states, hybrid parameters 88
5.5 Combined state and parameter estimation 92
5.5.1 Continuous states, continuous parameters 93
5.5.2 Discrete states, continuous parameters 95
5.5.3 Hybrid states, continuous parameters 98
5.5.4 Continuous states, discrete parameters 100
5.5.5 Discrete states, discrete parameters 100
5.5.6 Hybrid states, discrete parameters 101
5.5.7 Discrete states, hybrid parameters 101
5.6 Conclusions 101

6 Application: Hybrid Model-Parameter Estimation 103
6.1 Introduction 103
6.2 Previous Work 107
6.3 Bayesian estimation of the hybrid joint density 109
6.4 Interpretation and discussion 112
6.4.1 Kalman Filter variants 113
6.4.2 Particle Filter approaches 114
6.5 Implementation 115
6.5.1 Measurement equations 115
6.5.2 Explicit equations for Particle Filters 119
6.5.3 Uncertainty on the models 121
6.6 Experimental results 121
6.6.1 CF Estimation using only twist and wrench data . . 122
6.6.2 Fully hybrid estimation 126
6.7 Summary and Conclusions 128

7 BFL: The Bayesian Filtering Library 133
7.1 Introduction 133
7.2 Requirements 133
7.3 Other libraries 135
7.3.1 Bayes++ 135
7.3.2 Scene 135
7.3.3 Bayesian Networks software libraries 135
7.3.4 CES 136
7.4 Overview of BFL 136
7.4.1 Class interface design 136
7.4.2 Abstraction layers 137
7.4.3 Open Source software 138

7.5 Applications . . . 138
7.5.1 Mobile robot localisation . . . 138
7.5.2 Hybrid estimation . . . 140
7.5.3 Tracking a plane with an XY-platform . . . 140
7.6 Conclusions . . . 141

8 Conclusions **143**
8.1 Introduction: situation of the work . . . 143
8.2 Contributions . . . 144
8.2.1 Classification of Bayesian inference based on nature of the random variables . . . 144
8.2.2 Autonomous Compliant Motion . . . 145
8.2.3 New hybrid Bayesian models . . . 146
8.2.4 Software . . . 146
8.3 Limitations and future research . . . 148

References **151**

A The Expectation Maximization algorithm **169**

Index **171**

Curriculum Vitae **171**

List of Publications **I**

Nederlandse Samenvatting **I**
1 Inleiding . . . I
1.1 Situering . . . I
1.2 Een geïntegreerde aanpak van ACM . . . III
1.3 Bayesiaanse Waarschijnlijkheidleer . . . V
1.4 Filtering . . . VI
1.5 Bijdragen . . . VI
2 Monte Carlo methodes . . . VII
2.1 Monte Carlo algoritmes . . . VIII
2.2 Sequentiële Monte Carlo methodes . . . IX
3 Classificatie van Bayesiaanse algoritmes . . . X
4 Toepassing: hybride model-parameter estimatie bij assemblage . . . XIII
4.1 Probleemomschrijving . . . XIII
4.2 Hybride aanpak . . . XIV
4.3 Algoritmes . . . XV
4.4 Experimentele resultaten . . . XVI
5 BFL, een bibliotheek voor recursieve Bayesiaanse algoritmes XVI

6 Conclusies xviii
6.1 Bijdragen van dit werk xix
6.2 Beperkingen en toekomstig onderzoek xx

Chapter 1

Introduction

1.1 Introduction

Compliant motion in robotics refers to tasks in which the robotic end effector or a tool it holds moves *in contact* with the environment. Compliant motion allows robots to deal with uncertainties in the environment, in situations where purely position controlled strategies fail. *Autonomous compliant motion* (ACM) refers to situations in which the robot is able to autonomously accomplish a high-level task, by sensibly using its sensors to learn more about the environment and/or the tools/objects it holds.

Tasks involving autonomous compliant motion are ubiquitous: The ultimate domestic service robot robot should be able to open a door or a closet, whether it is closed or ajar. Our robotic household should be capable of clearing the table, no matter where we left our cups and dishes. Industrial applications include assembly tasks and the machining of workpieces in unstructured environments.

Figure 1.1 shows the ACM experiment that is used to demonstrate this research. A Kuka 361 Serial manipulator, equipped with a force-torque sensor, autonomously assembles a cube (the Manipulated Object MO) into a corner, formed by three perpendicular planes (the Environment Object E0). The unknown parameter variables this thesis considers are the location of the MO with respect to the end effector of the robot, and the location of the EO with respect to a fixed "world frame". These unknown variables are often referred to as the *geometrical parameters*. The measurements that are used for the estimation of these parameters are the wrenches (force-torque measurements) from the force sensor and the position measurements from the encoders of the robot. The obtained measurements yield information about the geometrical parameters via mathematical models expressing the relationship between

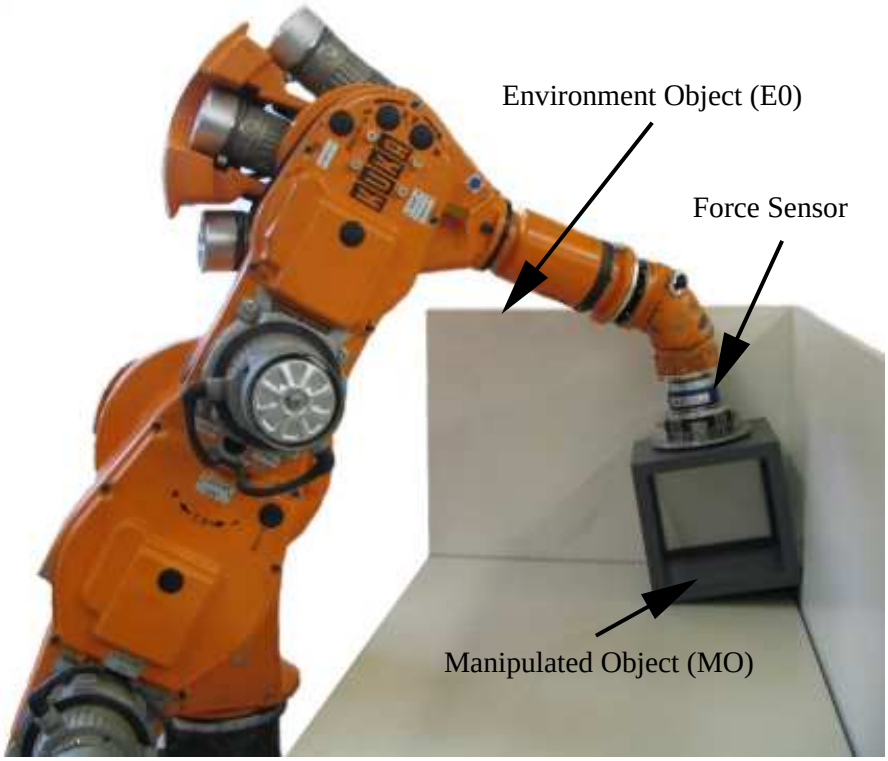


Figure 1.1: Execution of the "cube in corner" autonomous compliant motion experiment. A Kuka 361 Serial manipulator, equipped with a force-torque sensor, autonomously assembles a cube (the Manipulated Object MO) into a corner, formed by three perpendicular planes (the Environment Object E0). The unknown continuous parameter variables this thesis considers are the location of the MO with respect to the end effector of the robot, and the location of the EO with respect to a fixed "world frame". These are often referred to as the *geometrical parameters*. The measurements that are used for the estimation of these parameters are the wrenches (force-torque measurements) from the force sensor and the position measurements from the encoders of the robot.

parameters and measurements. These models are function of the discrete *Contact Formation* (CF). Indeed, as shown in figure 1.2, the execution of a

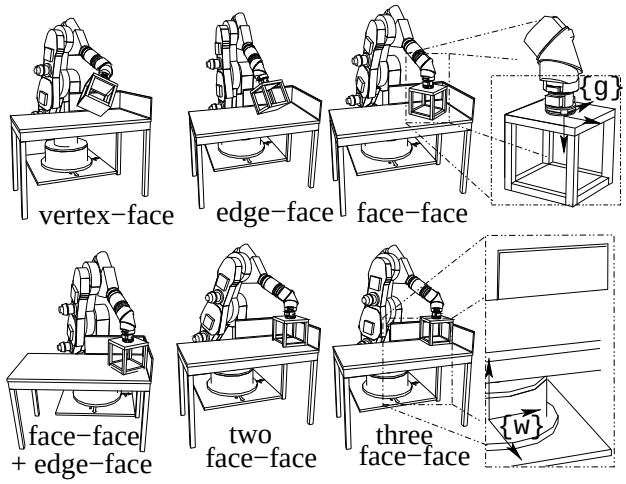


FIGURE 1.2: Execution of the "cube in corner" autonomous compliant motion experiment: different *Contact Formations* (CF).

typical task can be segmented into different *discrete* Contact Formations (CF, e.g. a *vertex-plane* contact, an *edge-plane* contact or a *plane-plane* contact): every CF gives rise to a different *measurement model*. In order to be able to estimate the geometrical parameters during the force-controlled execution of CF sequences, it is thus necessary to *recognize* the current CF. Furthermore, recognizing the current Contact Formation is necessary to select the appropriate control algorithm.[1]

The aim of this thesis was to check if Autonomous Compliant Motion applications in particular could benefit from a relatively new group of algorithms commonly called Particle Filters or Sequential Monte Carlo methods. These stochastic algorithms were discovered in the nineties as a useful alternative to the better known Kalman Filter variants in situations where non-linear models occur, especially in situations where ambiguity remains after gaining information.

During this PhD work, Lefebvre (2003) developed a new Kalman Filter variant, the Non-Minimal State Kalman Filter, which proved to be the optimal[2] choice for dealing with continuous parameter estimation, *given* these

[1] The problem of *CF recognition* includes the simpler problem of *CF transition detection.*

[2] We use the term "optimal" since the filter provides accurate parameter estimates recursively under large uncertainties with a limited computational cost and we believe it's the best choice for these applications.

parameters are observable and assuming no ambiguity remains when decisions have to be taken. However, the latter conditions only cover a subset of all estimation problems in general and in ACM in particular. This thesis shows that sequential Monte Carlo methods provide a valuable alternative in situations not covered by the NMSKF, i.e. where large uncertainties lead to *ambiguity* in CF belief, despite their higher computational complexity.

Previously, estimation research for ACM at this department focused on the estimation of geometrical parameters. This thesis additionally estimates the Contact Formation. This allows to deal with larger uncertainties on the geometrical parameters. The combination of the estimation of discrete Contact Formation states and continuous geometrical parameters leads to hybrid models, and poses new challenges to the estimators. Previous research only used consistency tests for detecting Contact Formation *transitions*, which leads to a large number of possibilities (and thus computational requirements) after transitions in case of large uncertainties. By *explicitly modeling* CF transitions, the next CF can be "predicted". This leads to more modest requirements for the estimator.

1.2 A system for ACM

Figure 1.3 shows the control scheme for the ACM system to be developed at our department. The autonomous control system at PMA consists of three components:

1. The controller component calculates the velocity setpoints sent to the robot's drives. The controller uses the hybrid control paradigm (Raibert and Craig 1981): Directions that allow free motion are velocity controlled, the others are force controlled. Therefore, the recognition of the current Contact Formation is important in order to choose the appropriate discrete control law. The inputs to this control law are the Contact Formation compliant path as specified by the planner component.

2. The estimator uses the force-torque and position measurements to provide estimates of the discrete Contact Formation and continuous geometrical parameters to the online plan execution component.

3. The online plan execution component transforms the high level task plan into force-velocity setpoints for the control. It takes into account the high level information from a task plan and the results from the estimator. Some work has already been done on off-line planning (Xiao and Ji 2001), off-line active sensing (Lefebvre, Bruyninckx, and De Schutter 2005b), and the preparation of the planning results for the controller component, denoted as *the compliant task generator* (Meeussen,

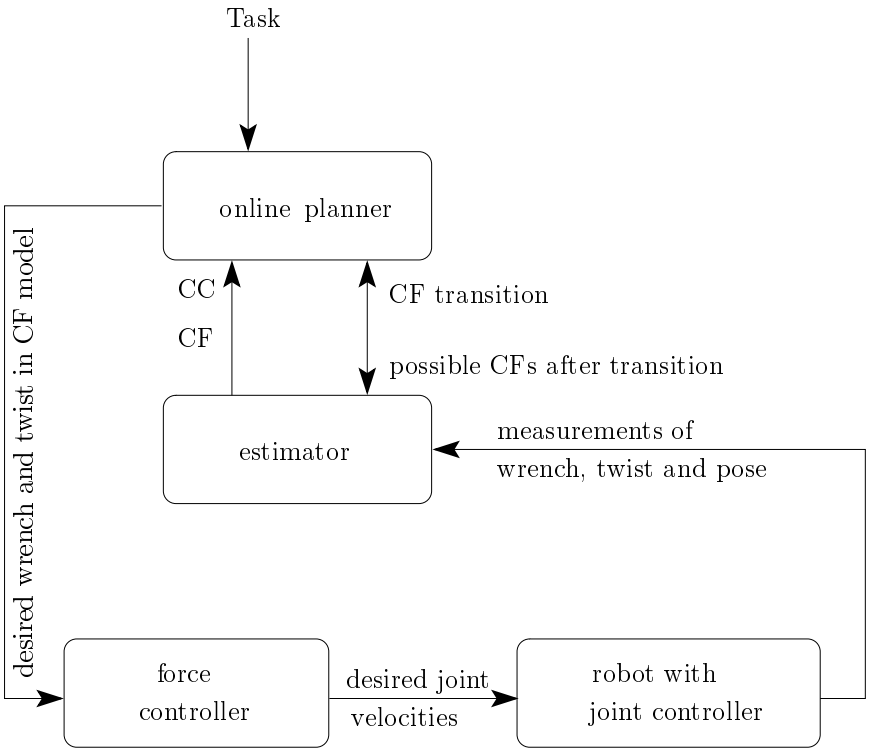


FIGURE 1.3: The "control" scheme of the ACM system at PMA, consisting of three interacting components: control, estimation and online planning. "CC" denotes Contact Configuration and corresponds to the estimated value of the geometrical parameters.

De Schutter, Bruyninckx, Xiao, and Staffetti 2005). The major hurdle to take here is to take into account online and recursively the information provided by the estimator, sometimes called *dynamic replanning*.

This work deals with the estimator component. It presents a strategy for allowing the estimator to deal with large uncertainties, such that ambiguity in CF recognition arises. By incrementally improving the accuracy of estimates of geometrical parameters and Contact Formations, the plan execution component can adapt its setpoints online to achieve a faster assembly process. The greatest challenge for future research is to decide how the online plan execution component should deal with ambiguity in CF estimates and large uncertainties on the geometrical parameter estimates.

Furthermore, as pointed out later on in this thesis, future ACM research can benefit from the use of *global* sensor information as provided by laser scanners or cameras. Indeed, the encoder information from the motors of the robot and the forces/torques from a force sensor provide only information about the current contact area between the robot and the environment, i.e. *local* information. Nowadays, cheap cameras are widely available and protocols such as (realtime versions of the) IEEE 1394 standard are available to process camera data in realtime.[3]

1.3 Applications of this research

The main ACM application and experiment dealt with in this thesis is the cube in corner assembly task illustrated in figure 1.1. However, the discussed approach is also valid for other applications dealing with simultaneous discrete model recognition and continuous parameter estimation.

Another application of this research is found in the autonomous machining of workpieces without the need for expensive, time-consuming fixtures. Such workpieces typically consist of several discrete geometrical primitives, such as planes, cylinders, etc. An example is shown in figure 1.4. CAD models of these workpieces describe the relative orientation of the primitives with respect to each other. If the robot can localize these workpieces to a desired accuracy (De Geeter, Van Brussel, De Schutter, and Decréton 1996), it can use the estimated parameters to perform machining operations. When localizing these workpieces under large uncertainties, it is necessary to recognize the current geometrical primitive (i.e. a discrete time-varying model) in order to estimate the location of the workpiece, since each geometrical primitive corresponds to a different measurement model.

[3]Note however, that not all sensors can be used in all environments such as in underwater operations.



FIGURE 1.4: Typically, workpieces to be machined consist of a number of primitive geometric models such as planes, cylinders, ... A model describing the relative orientation of these primitives is available from CAD models.

Recently, some work has been done about model *building* from a given set of primitives (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004) in a programming-by-demonstration context. Slaets tries to build a model from the cube-in-corner environment (figure 1.1) *without* using the notion of contact formation transitions. Each time the current measurements indicate that a new contact between the Manipulated Object and the Environment Object is established, a number of discrete model possibilities is compared to select the most probable model. The number of possibilities depends on information from the environment and the task to be executed, which is supposed to be known.

1.4 The Bayesian approach

Bayesian probability theory allows deterministic and optimal information processing under uncertainty in the sense that no information is created or deleted by Bayesian algorithms. Of course, practical trade-offs between information optimality and computational complexity must always be made, but this text identifies those trade-offs, and separates the generic trade-offs from the application dependent ones.

The discussion between supporters and opponents of Bayesian theory has not yet come to an end. Unfortunately, this discussion mostly boils down to the somewhat useless yes-or-no game of using prior information or not. This thesis is not the place for such a debate. However, my personal opinion on the matter is that Bayesian theory offers a unifying and consistent framework to reason about uncertainty, and perhaps more importantly: Bayesian methods force the researcher to make all implicit (modeling) assumptions explicit: Most (if not all) algorithms for estimation, can be shown to fit in the Bayesian framework with specific assumptions or simplifications. This has led to cross fertilization of research communities.

This thesis uses Bayesian probability theory to estimate unknown (sometimes denoted hidden or latent) variables. The term "rigorous" in its title refers to the fact that *all* unknown variables (i.c. Contact Formations and geometrical parameters in case of our experiments) are estimated by fully Bayesian methods, and that all trade-offs and assumptions made to obtain the results are made explicit in the text.

Model. One of the most frequent words in this thesis and Bayesian probability theory is probably the term model. Depending on the context, model can have different meanings.

Basically, the term model in this thesis denotes a mathematical, often simplified, description of a part of reality. We make a distinction between *explicit* and *implicit* models. The former is used to denote mathematical

models that use parameters which have a *physical* meaning. This allows for easier interpretation of the estimation results. However, this is not always possible for complex problems.

In the context of Bayes' rule, model refers to so-called probabilistic *likelihood* models, relating random variables with each other. This meaning of "model" *comprises* the first one, but adds the notion of uncertainty (probability) to the deterministic model. The random variables can be known or unknown. The *measurement model* relates known variables (data, in this thesis typically sensor measurements) to unknown variables, which can be parameters or states. *System models* only apply to dynamic systems, and relate the unknown state of the system at time $k+1$ to the state of the system at timestep k. Both these models can contain extra variables, either known (e.g. inputs to the system) or unknown (parameters). Sometimes, the inputs to the system are also unknown (e.g. when tracking a hostile missile). In that case, random noise is inserted into an otherwise static system. The random noise compensates for the unknown inputs.

Bayesian methods can be applied to implicit as well as to explicit models. Their greatest virtue is the fact that they often contribute in making the assumptions and approximations that are used in non-Bayesian algorithms explicit. So the term implicit on a mathematical modeling level doesn't mean that Bayesian methods are not applied.

Furthermore, the terms implicit and explicit are also used to describe equations. Whereas $\boldsymbol{y} = h(\boldsymbol{x})$ is denoted as an *explicit* equation, $h(\boldsymbol{x}, \boldsymbol{y}) = 0$ is called an implicit equation.

A third meaning of model in this work is the use of model as *graphical model*. A graphical model (or Bayesian Network) is a graphical representation of the set of probabilistic models used to describe a part of reality.

Under *model building*, we understand the estimation of a number of continuous parameters of a mathematical model. *Model selection* refers to the estimation of a discrete number of possible model choices.

1.5 Contributions

This thesis includes contributions in the field of Bayesian modeling, estimation for autonomous compliant motion, and software for recursive Bayesian estimation.

Bayesian Modeling This thesis provides new insights into the field of estimation based on the nature of the variables, and compares various algorithms and models from different research fields with respect to the assumptions they make. By considering a joint posterior density of discrete models and continuous states/parameters, Chapter 5 also shows how cross dependencies

between discrete (switching) model selection and continuous state/parameter estimation can be taken into account. This leads to new Bayesian models. We show why Kalman Filter techniques cannot perform inference in these models and apply sequential Monte Carlo techniques to estimate the joint posterior density. The classification of Bayesian algoritms provided in Chapter 5 also allows non-Bayesian but application domain experts to define their needs and choose Bayesian algorithms without "wasting" their time finding their way in the "chaos" of available Bayesian models and algorithms.

Autonomous Compliant Motion Chapter 6 focuses on the estimation part of an autonomous compliant motion system. For ACM problems, there's a dependency between the estimation of continuous geometrical parameters and discrete switching CFs. The estimator in this thesis uses a rigorous Bayesian approach to take into account this dependency, using fully Bayesian methods to estimate both the unknown geometrical parameters and Contact Formations. This allows the estimator to tackle larger uncertainties than was previously possible (Gadeyne, Lefebvre, and Bruyninckx 2005).

The joint posterior density of geometrical parameters and discrete contact formations is estimated recursively via Sequential Monte Carlo techniques (commonly denoted as Particle Filters). This requires the use of explicit measurement models. Existing implicit measurement equations from previous research are transformed into explicit ones that can be used in Particle Filters.

This thesis also provides some hints on how to take into account uncertainty when solving the active sensing problem.

Software The recursive Bayesian estimation of discrete, continuous or hybrid posterior densities requires a flexible software estimation framework. The *Bayesian Filtering Library* (Gadeyne 2001b) (BFL, Chapter 7) is an open source software project that fulfils these requirements and is independent of a particular research field. It has been used for several estimation jobs in ACM but also for mobile robotic localisation and tracking objects with cameras. BFL has been integrated into the newly developed control software Orocos (Soetens and Bruyninckx 2005), which presents an important step towards the fully autonomous compliant motion system of figure 1.3.

1.6 Overview of this thesis

Chapter 2 introduces the concept of *recursive* Bayesian inference, and the notion of the joint posterior over parameters and states. The applications in this thesis require updates of the posterior in bounded time and memory. This leads to the concept of *filtering*. The Chapter also describes Bayesian

Networks (BNs), which offer automated inference for non-recursive applications, and Dynamic Bayesian Networks (BNs for recursive inference). BNs are important to study similarities and differences between different stochastic models and algorithms. Finally Section 2.7 presents some hints on how future research on online intelligent plan execution might deal with ambiguity in a Bayesian context.

Bayes' rule does not specify *how to represent* the posterior. While previous research at this department focused on analytic representations, Chapter 3 describes what Monte Carlo methods are and how they can be used to represent the posterior density as a discrete set of random samples. The focus of the Chapter is on methods that are used in the remainder of this thesis and in BFL.

Particle Filters, also known as Sequential Monte Carlo methods (Chapter 4) apply Monte Carlo methods in the context of recursive filtering. The Chapter provides an overview of this relatively new field. Particle Filters have been shown to generate promising results in the field of non-linear filtering, where analytical filters typically fail. However, due to the fact they represent the posterior as a finite number of samples, numerical problems often arise and prohibit fool proof application of the basic Sequential Importance Sampling algorithm. Various methods to solve these numerical issues are discussed.

An enormous number of algorithms for performing Bayesian inference has been developed during the last decades. Most of them are limited to a certain subclass of Bayesian models. Some of them can be applied online, others need an off-line *training* period and/or approximate the posterior PDF by a single value. Chapter 5 provides a comprehensive overview of these algorithms, based on the nature of the unknown random variables in the joint posterior. This exhaustive approach leads to the discovery of some new Bayesian models.

One of these models, a Hidden Markov Model where the unknown continuous parameters can influence the evolution of the discrete states, is used for the simultaneous estimation of geometrical parameters and CFs during a cube-in-corner assembly task in Chapter 6. Analytic filters cannot be applied for online inference in such models, but Particle Filters can solve this estimation problem, however at a higher computational cost. The new model allows to deal with larger uncertainties during ACM tasks.

The overview of estimation algorithms in Chapter 5 also calls for a flexible software framework. Chapter 7 describes the Bayesian Filtering Library (BFL), an open source C^{++} software library for fully recursive Bayesian filtering.

This work only presents one step forward in the building of a fully autonomous compliant motion system. Chapter 8 summarizes the contributions of this work and provides some hints for future research.

Chapter 2

Bayesian Inference

2.1 Introduction

This Chapter provides the Bayesian foundations of the algorithms used in this thesis, and particularly the Bayesian background of the sequential Monte Carlo algorithms presented in Chapter 4. It is complementary to the literature survey presented by (Lefebvre 2003). In that respect, a detailed literature survey on hypothesis testing and information measures is omitted from this Chapter. Neither does it rediscuss the merits of Bayesian theory with respect to other approaches of dealing with uncertainty, or the choice of a prior distribution. To find out more about the latter, see e.g. (Lindley 1972; Jaynes 1996).

One of the merits of this thesis is that it solves a previously unsolved *parameter/state estimation* problem—this term includes many terms often found in literature: model selection, hypothesis testing, learning, filtering, smoothing, ...[1]—with a rigorous Bayesian approach. Therefore, this Chapter starts with the definition of the joint posterior (Section 2.2), that will be used in the rest of this thesis. Section 2.3 describes some of the models used in this thesis based on that particular joint posterior. Due to the constant memory and time requirements of *online* estimation, most of the time, a marginal of the joint posterior is used (Section 2.4). Section 2.5 details the inference process by means of Bayes' rule. The link with Bayesian network theory is made in Section 2.6, and Section 2.7 shortly resumes the issue of *decision making*, which is more thoroughly discussed in (Lefebvre 2003).

[1] Chapter 5 deals with the similarities, differences and connections between all those terms.

2.2 The joint posterior

In a Bayesian perspective, all problems dealt with in this thesis boil down to the estimation of a joint posterior PDF of unknown[2] Random Variables $\boldsymbol{X}$ and $\boldsymbol{\Theta}$:

$$P(\boldsymbol{X}_{1:k} = \boldsymbol{x}_{1:k}, \boldsymbol{\Theta} = \boldsymbol{\theta} | \boldsymbol{Z}_{1:k} = \boldsymbol{z}_{1:k}). \tag{2.1}$$

Some background on this notation:

- We make an explicit distinction between *states* $\boldsymbol{X}$ and *parameters* $\boldsymbol{\Theta}$. In the rest of this thesis, we consider a parameter variable to be a stochastic variable that does not change over time *in the particular model* used during estimation. This distinction does not necessarily correspond to the physical reality of the problem. E.g. the length of a steel bar could vary during the experiment due to temperature fluctuations, but if this temperature dependency is not modeled, we consider the bar length to be a parameter of the posterior.

  Note that most authors do not consider parameters and states separately and use one vector valued random variable for modeling:

  $$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_k \\ \boldsymbol{\theta} \end{bmatrix}. \tag{2.2}$$

  However, this thesis *explicitly* distinguishes between parameters and states. The main reason for this is the limited applicability of some of the presented algorithms to either state or parameter estimation, e.g. using a standard Particle Filter algorithm for the estimation of parameters will mostly result in numerical instability. Chapter 5 discusses the classification of the joint posterior based on the presence and nature of $\boldsymbol{X}$ and $\boldsymbol{\Theta}$ thoroughly.

  Sometimes, it is useful to split $\boldsymbol{\Theta}$ or $\boldsymbol{X}$ up into parts $\boldsymbol{\Theta}^i$ or $\boldsymbol{X}^i$, since those parts can have different conditional independence relationships with other random variables in the model. This is explained in depth in Section 2.6. This also has consequences on algorithmic level. One of the virtues of Bayesian Networks (Section 2.6) is that they can clarify the relationship between the choice of models and the complexity of algorithm implementations.

- Vectors are denoted in **boldface**.

[2] Other frequently used terms include *hidden* variables or *latent* variables.

- Wherever the distinction between a *stochastic variable* (also denoted as random variable, RV) $\boldsymbol{A}$ and a *particular value* $\boldsymbol{a}$ of that stochastic variable is clear, we omit the uppercase symbol. E.g. eq. (2.1) is mostly written as
$$P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k}), \tag{2.3}$$
which reads as "the probability that the state variable $\boldsymbol{X}_{1:k}$ has the value $\boldsymbol{x}_{1:k}$, and the model parameter $\boldsymbol{\Theta}$ has the value $\boldsymbol{\theta}$, given that the measurement (data) values $\boldsymbol{z}_{1:k}$ are known.
- The subscript $A_{1:k}$ denotes the range of values $A_1, \ldots, A_k$. A_j denotes a state random variable (which is dynamic) at time j.

2.3 Applications

Eq. (2.1) can represent a lot of models. Both $\boldsymbol{\Theta}$ and $\boldsymbol{X}$ can be either discrete, continuous or hybrid. Sometimes either the parameter or state vector are assumed to be known. This Section discusses some examples of applications used in this thesis or in the autonomous compliant motion research related to this thesis. It is by no means meant to be an exhaustive enumeration, but it should help in better understanding the rest of this text, without having read later chapters. Chapter 5 discusses this joint posterior in more detail, and also focuses on the representation of this posterior and the consequences of that choice on an algorithmic level.

The autonomous compliant motion (ACM) applications described in Section 1.3 deal with *static* localization. The parameter vector $\boldsymbol{\theta}$ always contains a continuous part, i.c. the geometrical parameter vector. This is due to the fact that the uncertainty on those geometrical parameters is typically much larger than the uncertainty on the (dynamic) position of a calibrated serial manipulator. The latter is therefore assumed to be exactly known, although the presented algorithms could also take this uncertainty into account. This would result in a 6D-SLAM problem, with an increased complexity as a consequence. Mekhnacha, Mazer, and Bessière (2001) take the geometrical uncertainties on the robot kinematics explicitly into account, and apply off-line Bayesian reasoning to obtain Maximum Likelihood values for the joint angles of serial manipulators, given certain geometrical constraints. The measurements used for the estimation are wrenches (forces and torques) from a wrist force sensor, and position measurements from the encoders.

Another application is the online calibration of a serial manipulator: in that case the geometrical parameter vector is extended with other parameters. These parameters can have a physical meaning, e.g. the length of the different links of the manipulator, or they can be used to characterize the uncertainty of a part of the motion. E.g. given a certain joint angle α_k and velocity ω_k at

timestep k, what is the resulting uncertainty $P(\alpha_{k+1})$ on the joint angle at timestep $k+1$, if additive Gaussian uncertainty is considered:

$$P(\alpha_{k+1}) = \mathcal{N}\left(\alpha_k + \omega_k \Delta t, \sigma\right),$$

where σ is considered unknown and to be estimated during calibration. In a Bayesian context, the latter application (when parameters have no physical meaning and are often denoted as *hyperparameters*), is typically denoted as *hierarchical Bayes modeling* or *hierarchical Bayes* in short.

In the case of combined parameter and CF-estimation (Chapter 6), $\boldsymbol{\Theta}$ contains the continuous geometrical parameter vector and $\boldsymbol{X}$ contains the discrete Contact Formation. In the case of (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004), $\boldsymbol{\Theta}$ is a hybrid vector containing both the continuous geometrical parameter and a discrete variable. This is a example of combined parameter estimation and hypothesis (model) testing.

Eq. (2.1) is also applicable to many problems in mobile robotics research (such as global localization and SLAM (Dissanayake, Newman, Clark, Durrant-Whyte, and Csorba 2001)), combined localization and intention estimation for shared wheelchair control (Demeester, Nuttin, Vanhooydonck, and Van Brussel 2003; Vanhooydonck, Demeester, Nuttin, and Van Brussel 2003), fault diagnosis (Mehra and Peschon 1971; Verman, Gordon, Simmons, and Thrun 2004; De Freitas, Dearden, Hutter, Morales-Menéndez, Mutch, and Poole 2004), ...

2.4 Filtering

Most often, we're not interested in the values of *all* the unknown variables of eq. (2.3). Typically, we don't want to know the values of the past states $\boldsymbol{x}_{1:k-1}$. This leads to a posterior of the following form:

$$P(\boldsymbol{x}_k, \boldsymbol{\theta} | \boldsymbol{z}_{1:k}). \tag{2.4}$$

In case of a discrete state vector $\boldsymbol{x}$, Eq. (2.4) is obtained via a summation over the unknown past states, if $\boldsymbol{x}$ is continuous, this posterior density is obtained by integration of the joint posterior.[3] This process of eliminating certain variables by integration or sommation is often called the *marginalization* of the joint posterior (2.3) and Eq. (2.4) is said to be one of the *marginals* of the joint posterior $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k})$.

The estimation of the PDF described by eq. (2.4) is denoted as *filtering*. In case the posterior is $P(\boldsymbol{x}_j, \boldsymbol{\theta} | \boldsymbol{z}_{1:k})$, literature uses the terms *smoothing* if $j < k$ and *prediction* if $j > k$. Special cases also occur when there are only parameters *or* states to be estimated.

[3] Or a combination of both if $\boldsymbol{x}$ is hybrid.

Filtering is the most important application in this thesis: the goal of the estimation problems dealt with in this work is the *recursive* estimation of the joint posterior: measurements are taken on-line, and the update of the posterior should be incremental. Indeed, as the *information* about the unknown variables increases, the robot can take actions in order to acquire measurements that will maximize the information gain. The more information the robot has gathered about the unknown variables, the more accurately it can choose its actions in order to gain maximum information. This optimization process is called *active sensing* (Bajcsy 1988; Kaelbling, Littman, and Cassandra 1998; Kröse and Bunschoten 1999; Lefebvre 2003) and discussed more thoroughly in Section 2.7. *Online* active sensing is not possible when processing the measurements *in batch*. Therefore, this thesis focuses on algorithms that perform the update from the posterior at timestep $k-1$ to k in

- a bounded time interval;
- a bounded memory footprint.

Both requirements are necessary in order to use the estimation algorithms online during realtime controlled tasks, such as the assembly task with a serial manipulator described in Chapter 6. These requirements are also fulfilled in the software framework developed in this thesis (Chapter 7). The estimation of the *joint* posterior (2.3) always violates the second requirement, therefore marginalization will be performed at each timestep.

2.5 Bayes' rule

To get a recursive description of the posterior (2.3), Bayes' rule and the product rule[4] are applied:

$$\begin{aligned} P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k}) &=^1 \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_{1:k}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k-1}) P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k-1})}{P(\boldsymbol{z}_k | \boldsymbol{z}_{1:k-1})} \\ &=^2 \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k-1})}{P(\boldsymbol{z}_k | \boldsymbol{z}_{1:k-1})} \\ &=^3 \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k-1}) P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k-1})}{P(\boldsymbol{z}_k | \boldsymbol{z}_{1:k-1})} \\ &=^4 \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k-1})}{P(\boldsymbol{z}_k | \boldsymbol{z}_{1:k-1})}. \end{aligned} \tag{2.5}$$

[4] Bayes' rule and the product rule are in fact the same since one can be derived from the other: $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$.

1 is a straightforward application of Bayes' rule, 3 of the product rule. Both 2 and 4 are applications of the *Markov assumption* (Chung 1960; Howard 1960), which is an assumption about the model necessary to allow recursive calculation of the posterior as described in the previous Section. It states that knowledge about the current states and parameters suffices to make predictions about future states and measurements.[5]

So for any recursive combined parameter/state estimation problem for which the Markov assumption holds, we can write the posterior at timestep k as:

$$P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}), \tag{2.6}$$

or, in words

$$posterior = \frac{likelihood}{evidence} \times prior. \tag{2.7}$$

$P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$, often denoted as *evidence*, is independent of both states and parameters and can hence be considered a *normalization* factor to ensure the posterior is a true PDF (i.e. it integrates (continuous) or sums up (discrete) to 1).

The *likelihood* function $P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})$ is a PDF over the stochastic variable $\boldsymbol{z}_k$ and a function of $\boldsymbol{z_k}, \boldsymbol{x}_k, \boldsymbol{\theta}$. It is often denoted as the *measurement model*: it predicts the measurement from the current state and parameter. The other likelihood function, $P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})$ is called *system model* in the rest of this thesis: it predicts the next state from the current state and parameter. Lefebvre (2003), p. 36 describes the one-to-one relationship between a state-space model and these likelihood functions for the particular case of a linear state space model with additive Gaussian uncertainty on both system and measurement model. This mapping is always possible for Markovian models.

Both likelihood functions can contain extra variables. For the system model, these are mostly called *inputs* and denoted as $\boldsymbol{u}_{1:k} = \boldsymbol{u}_k \dots \boldsymbol{u}_k$. Examples are the velocities (Cartesian or joint) sent to the motor controllers of a robot. In case of the measurement model, the parameters are mostly called *sensor parameters* and denoted as $\boldsymbol{s}_{1:k} = \boldsymbol{s}_1 \dots \boldsymbol{s}_k$. E.g. in case of a mobile robot equipped with a radial ("Sick-like") laser scanner, the angle of the laser beam could be a sensor parameter. During the rest of this thesis, both types of parameters are only explicitly mentioned in the equations where ambiguity could arise (e.g. where simplified models are used, see Chapter 6) or when the extra variables are of utter importance, as in active sensing, where the goal is to choose the values of these parameters in order to optimize the information gain (Section 2.7).

[5] No predictions about parameters since, once they are known, they can't change anymore, by definition.

In the case of filtering and fully continuous variables, eq. (2.6) becomes

$$P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})}{P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \int P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1}, \tag{2.8}$$

which is often denoted as a two-step *system* and *measurement* update:

$$P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}) = \int P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1}; \tag{2.9}$$

$$P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})}{P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}). \tag{2.10}$$

The *prediction step* calculates our belief over the state and parameter vector at timestep k, given the measurements until timestep $k-1$. It uses the posterior PDF at timestep $k-1$, and the system model to obtain the joint PDF $P(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}) = P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1})$. After that, $\boldsymbol{x}_{k-1}$ is marginalized out. The *correction step* takes into account the information provided by the new measurement $\boldsymbol{z}_k$ via the measurement model. Similar equations hold for discrete or hybrid systems.

2.5.1 Maximum Likelihood and Maximum A Posteriori

Figure 2.1 illustrates Bayesian inference for the particular case of the estimation of a one-dimensional parameter, given a Gaussian prior density and a Gaussian likelihood.

Sometimes, there are not enough resources to perform full Bayesian inference. In that case, approximative methods often replace PDFs of the unknown random variables by *point estimates*. E.g. suppose we want to describe the likelihood of a certain measurement $\boldsymbol{z}$, caused by an unknown parameter vector $\boldsymbol{\theta}$. The Bayesian probability of the measurement is

$$P(\boldsymbol{z}) = \int_{\boldsymbol{\theta}} P(\boldsymbol{z}|\boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{2.11}$$

We can approximate this probability by using the point estimate $\boldsymbol{\theta}^*$ instead of the full PDF:

$$P(\boldsymbol{z}) \approx P(\boldsymbol{z}|\boldsymbol{\theta}^*). \tag{2.12}$$

Furthermore, as discussed in Section 2.7, in many applications, the information present in the posterior PDF has to be reduced into a single value that is used in another algorithm, e.g. a deterministic control algorithm.

The *Maximum A Posteriori* (MAP) and *Maximum Likelihood* (ML) estimates are often used to reduce a PDF into such a point estimate. As can be seen from figure 2.1, the MAP estimate takes into account *prior* information,

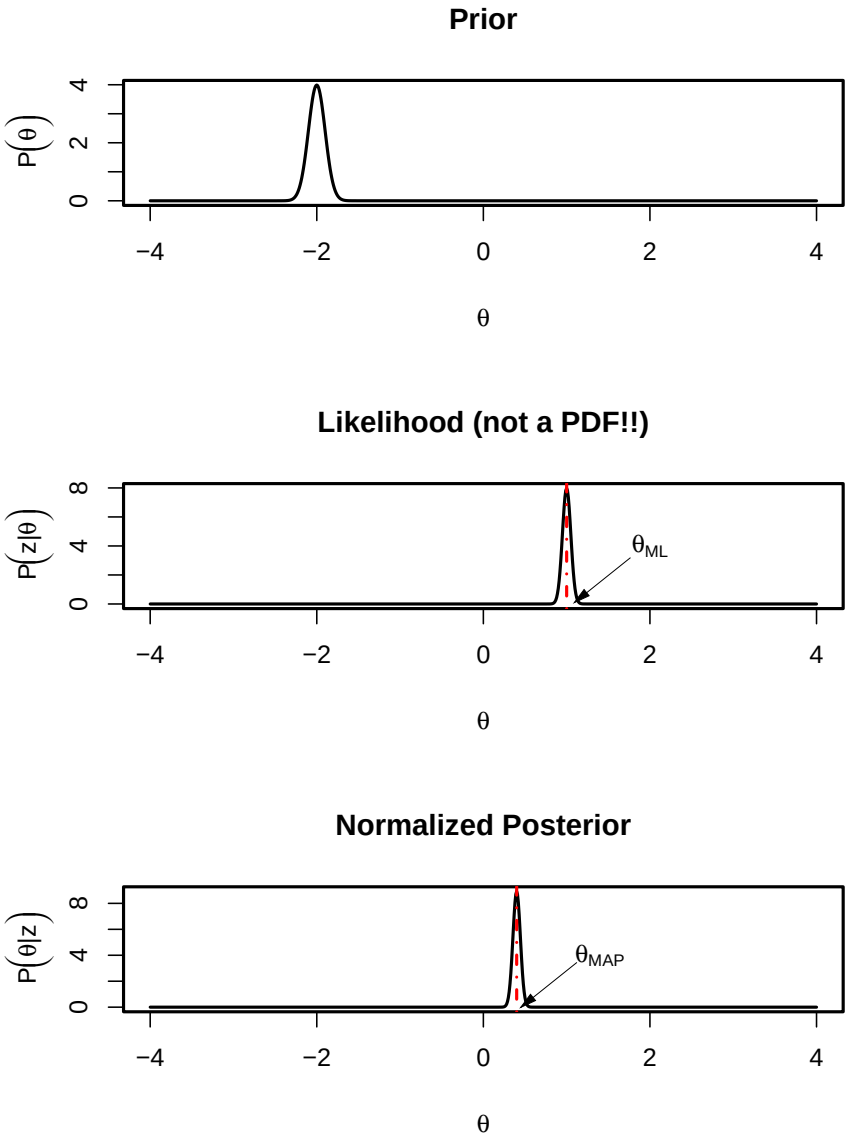


Figure 2.1: Bayesian inference for a 1D parameter estimation problem. θ_{ML} denotes the Maximum Likelihood (ML) estimate, θ_{MAP} the Maximum A Posteriori (MAP) estimate.

while the ML estimate only considers the information present in the measurements (which is equivalent to assuming a uniform prior). More precisely, the MAP estimate is the maximum of the posterior PDF $P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$—which is a *PDF* over the RVs $\boldsymbol{x}_k, \boldsymbol{\theta}$—, while the Maximum Likelihood estimate is the maximum over the likelihood *function* $P(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k, \boldsymbol{\theta})$—which is a *function* of $\boldsymbol{x}_k, \boldsymbol{\theta}$. Examples of such algorithms, such as the *Expectation-Maximization* (EM) algorithm (Dempster, Laird, and Rubin 1977), or the *Bayes' Point Machine* (Watkin 1993), are discussed in Chapter 5.

2.5.2 Conjugacy

Conjugacy (Diaconis and Ylvisaker 1979) is an interesting property when performing Bayesian inference, since it singles out classes of PDF models for which inference can be executed very fast. For a certain likelihood function, a family/class of *analytical* PDFs is said to be the conjugate family of that likelihood, if the posterior belongs to the same (PDF-)family. The best known example is when the likelihood is a normal distribution with known covariance matrix:

$$\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{known}).$$

Then any prior $\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ will yield a Gaussian posterior (figure 2.1 illustrates this). Together with the fact that a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, transformed through a linear transformation with matrix A, is still a Gaussian $\mathcal{N}(\boldsymbol{A\mu}, \boldsymbol{A'\Sigma A})$, this forms the Bayesian foundations of the Kalman Filter algorithm (Kalman 1960). Conjugate densities (see e.g. (Robert and Casella 1999), p. 31 for a list) are often used by Bayesians for purely computational reasons, even though they do not always correctly reflect the a priori belief.

For multi-parameter problems,[6] conjugate families are very hard to find, but many multi-parameter problems do exhibit *conditional conjugacy*. This means that the joint posterior itself has a very complicated form (and is thus hard to sample from) but its conditionals have nice simple forms. This property is extensively used in some sampling algorithms, typically in Gibbs sampling (Section 3.7.2).

2.6 (Graphical) Modeling—Bayesian Networks

States $\boldsymbol{X}$ as well as parameters $\boldsymbol{\Theta}$ in the joint posterior $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$ can be *discrete*, *continuous* or *hybrid* (i.e., having discrete as well as continuous components). E.g. the problem of simultaneous CF recognition and geometrical parameter estimation, dealt with in Chapter 6.1, has discrete states (the

[6] With multi-parameter we mean that the state vector $\boldsymbol{x}$ and/or the parameter vector $\boldsymbol{\theta}$ consists of several distinct physical or hyperparameters.

type of contact formation) and continuous parameters (the value of the geometrical parameters in the contact). Choosing the appropriate model is one of the most important tasks when solving an estimation problem. Indeed, the chosen model determines to a great extent the complexity of the algorithms necessary to estimate the posterior density.

A graphical and systematic approach of dealing with modeling and inference, that originates from the Artificial Intelligence research community are Bayesian Networks. A *Bayesian network* (Charniak 1991; Jordan 1999) (BN, also denoted *belief network*) provides a graph(ical) model representation for the joint distribution of a set of random variables, by means of a *directed acyclic graph* (DAG). Figure 2.2 shows an example and some terminology. The orientation of the arrows in this graph represents causality (and,

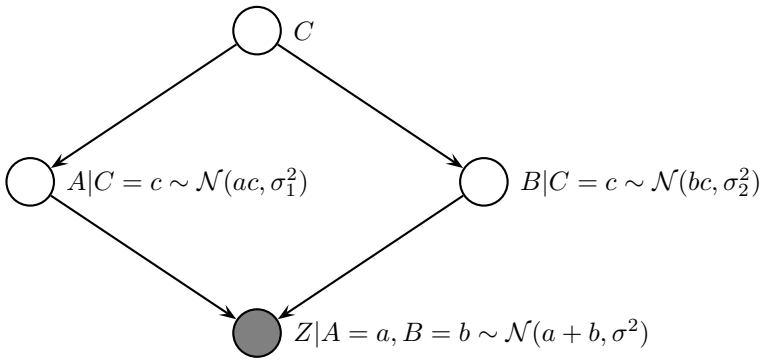


Figure 2.2: Example of a BN. Circles denote random variables (RVs). Grey circles denote observed RVs, transparent circles denote unknown (hidden) RVs. A and B are the *parents* of Z (which is the *child* of its parents).

therefore, the graph is acyclic). This relationship is denoted as a conditional (in)dependence, and is expressed by a Condition Probability Density (CPD). Conditional independence properties (often denoted as $A \perp\!\!\!\perp B \mid C$: A is independent of B given C) are important, because they can be used to simplify the general factorization formula for the joint probability. Any joint PDF can be factorized using the product rule

$$P(RV^1, \ldots, RV^n) = P(RV^1 \mid RV^2 \ldots RV^n) \ldots P(RV^{n-1} \mid RV^n) P(RV^n). \tag{2.13}$$

where RV denotes a Random Variable and the superscript ^{i} does not (necessarily) imply time dependence here: RV^1 and RV^2 could be two different

parameters, or $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the context of a state estimation problem. So, for the network of figure 2.2, this means

$$P(A, B, C, Z) = P(Z \mid A, B, C)P(A \mid B, C)P(B \mid C)P(C). \tag{2.14}$$

Conditional independence relationships lead to a recursive factorization form of eq. (2.13):

$$P(RV^1, \ldots, RV^n) = \prod_{i=1}^{n} P(RV^i | parents(RV^i)), \tag{2.15}$$

where the product is taken over all nodes i. This is sometimes called the *Directed Markov property*: any variable in the graph is conditionally independent of its *non-descendents*[7] given its parents:

$$RV \perp\!\!\!\perp nd(RV) \mid parents(RV),$$

where $nd(RV)$ denotes the non-descendents of RV. Again, for the network in figure 2.2, this leads to

$$P(A, B, C, Z) = P(Z \mid A, B)P(A \mid C)P(B \mid C)P(C), \tag{2.16}$$

since A is conditionally independent of B, given C, and Z is conditionally independent of C, given A and B.

"Standard" Bayesian networks are mostly used in clinical research (e.g. medical diagnosis (Cowell, Dawid, Lauritzen, and Spiegelhalter 1999)) and for artificial intelligence problems (e.g. expert systems (Lauritzen and Spiegelhalter 1988)), where there are hundreds of (often univariate, discrete and static) random variables and the full recursive factorization (2.13) would require an enormous amount of computation time and memory. Therefore, methods have been developed to systematically perform inference in such networks, exploiting the *Directed Markov property*. The general principles of *variable elimination* (Zhang and Poole 1996) and the *junction tree* algorithm allow to perform inference in such a network as efficiently as possible. Depending on the node types (discrete or continuous), the inference (exact or approximate), and the graph topology, a variety of algorithms exist (see e.g. (Jordan 1999; Murphy 2004a) or (Murphy 2002b), appendix B for an overview).

An example of the estimation of the joint posterior (2.1) used in this thesis is shown with the DAG of figure 2.3.[8] This is called a *Dynamic Bayesian network* (DBN) (Murphy 2002a; Murphy 2002b; Dean and Kanazawa 1989), due to the presence of unknown *state* variables, that change over time according

[7] Descendents is a synonym for children.

[8] Depending on the used models, some arrows might not be present.

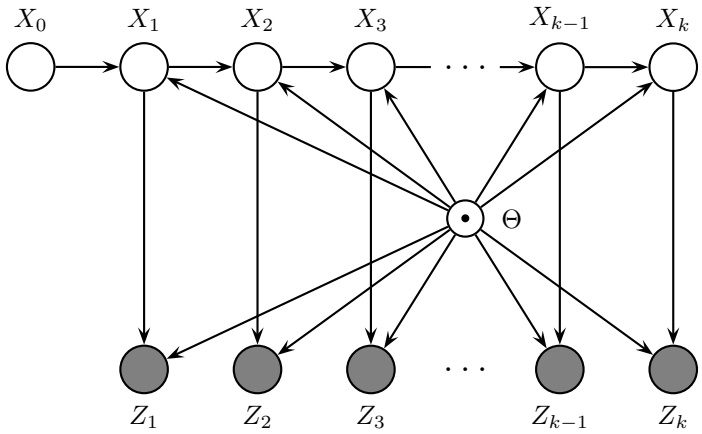


Figure 2.3: Bayesian network representing the joint posterior in its most general form (compare with eq. (2.6)). The dotted circle (Θ) denotes a parameter vector (i.e. all hidden RVs that do not change over time in the particular (BN) model used), empty circles denote states (i.e. hidden RVs that change over time according to a certain *system* model) and grey circles denote observed variables such as measurements. Inputs or sensor parameters are not explicitly mentioned in this figure.

to a system model. Well known examples of DBNs are Hidden Markov Models (HMM) (Rabiner 1989), if the state variables are discrete and Kalman Filter Models (KFM), where the CPDs express a linear relationship with additive Gaussian uncertainty:

$$P(\boldsymbol{RV}^i|parents(\boldsymbol{RV}^i)) = \mathcal{N}\left(\sum_j \boldsymbol{A}_j parent_j(\boldsymbol{RV}^i), \boldsymbol{\Sigma}\right), \tag{2.17}$$

where each $\boldsymbol{A}_j$ is a matrix. Unfortunately, the general junction tree algorithm[9] or one of its variants for performing inference in DBNs, is too resource intensive. Therefore, specific algorithms have to be used, which are only applicable to a certain subclass of DBNs. The *Viterbi* algorithm for HMMs and the Kalman Filter for KFMs are well-known examples. Chapter 5 describes these subclasses and the related algorithms in detail.

An important property of DBNs is the fact that they are able to represent *any* model-based analysis. Murphy (2002b) provides an excellent overview of the field of sequential data modeling. He also provides an overview of Bayes Network software (Murphy 2004b) and maintains the Bayes Net Toolbox (BNT) (Murphy 2001), an open source Matlab toolbox providing support for numerous BN inference algorithms. Based on BNT, Intel released OpenPNL (Intel) in 2003, an open source C^{++} library for inference in BN, which has no support yet for the DBN algorithms needed in this thesis.[10] Ongoing efforts focus on providing graphical model support for the statistical language *R* (gR).

Chapter 5 discusses the implementation of inference on the DBN described by eq. (2.1) in detail, with a particular focus on the application of sequential Monte Carlo techniques for these systems. Chapter 7 describes BFL, a C^{++} software package developed in this thesis to perform *online* fully Bayesian inference in DBNs.

2.7 Decision making

The result of Bayesian inference is always a PDF, expressing an updated (a posteriori) belief about the unknown random variables. That PDF can be represented by a number of particles, or described by an analytical function.

Why? As discussed in Chapter 1 and shown in figure 1.3, the estimator is only *one* part of an autonomous robot, and it needs to interact with a controller and a task planner component.

The controller component typically consists of a deterministic control algorithm, only accepting *deterministic* inputs. Therefore, the information that

[9] Often denoted as *unrolled junction tree* in the case of DBNs.
[10] Last checked 05/01/2005.

is contained in the posterior PDF has to be reduced to one or a few characteristics that can be used for further processing. This process is a particular case of *decision making*.

On the other hand, the information of the posterior PDF is also used by the planner. In a "fully autonomous" robot, the planner should be capable of executing a high level task, optimizing some objective function and taking into account one or several constraints. Active sensing[11] deals with this subproblem in order to reduce the uncertainty on the geometrical parameter estimate. In a Bayesian context, active sensing is mostly modeled by a Partially Observable Markov Decision Process (POMDP (Sondik 1971; Simmons and Koenig 1995; Cassandra 1998; Murphy 2000)). However, even for the smallest problems, online active sensing is not possible in realtime. Therefore, typically active sensing algorithms are also based on characteristics of the posterior.

How? If the posterior PDF only contains discrete parameters $\boldsymbol{\Theta}$—inference is then often referred to as *(static) hypothesis testing* or *(static) model selection*[12]—, most of the time the PDF is reduced to the single value $ArgMax(\boldsymbol{\theta})$.

If the posterior consists of a discrete dynamic part $\boldsymbol{x}_k$, the inference process is often referred to as switching model selection ((Lefebvre 2003), Section 3.4.2). An example is found in Chapter 6 which deals with the estimation of the sequence of CFs during the cube-in-corner assembly task: To be able to select a control algorithm, only one CF has to be selected from the discrete marginal posterior $P(CF_k|\boldsymbol{z}_{1:k})$.

For continuous distributions, the reduction process is mostly even harder, unless, for example, in the case of linear systems where the KF model applies and the posterior is fully represented by its mean and covariance matrix. Lefebvre (2003) discusses several *information measures* and provides a detailed comparison of the difference between choosing the *entropy* and the *covariance matrix*. Since, for autonomous compliant motion problems, we are ultimately interested in a single estimate, (Lefebvre 2003; Lefebvre, Bruyninckx, and De Schutter 2005b) suggests to use (a loss function of) the covariance matrix as a measure of uncertainty. However, when performing *active sensing* for autonomous compliant motion under large uncertainty, it might be useful to know if the large covariance matrix is caused by global uncertainty (i.e. a unimodal posterior with a large covariance matrix), or by some ambiguity (i.e. a multimodal posterior density, with a small entropy). Indeed, an (ad hoc) active sensing strategy might use *both* information measures in order to calculate the best possible action.

[11] Sometimes referred to as *optimal experiment design* or *reinforcement learning*.

[12] (Lefebvre 2003), Section 3.4.1 discusses several Bayesian possibilities for inference in such cases, one of which is applied in (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004).

2.8 Conclusions

This Chapter provides the foundations of all algorithms dealt with in this thesis. The Bayesian approach allows us to incorporate the information from different sensors recursively in an easy and theoretically sound way. Although the focus of this thesis is on sequential Monte Carlo methods (Chapter 4), the Bayesian background and modeling is crucial, since a change in the modeling stage can imply that other Bayesian algorithms become more appropriate to solve the estimation problem. Furthermore, due to the fundamental property of Bayesian theory that no information is created or deleted by Bayesian algorithms, it is the Bayesian framework in which sensor fusion techniques fit in "naturally".

Contrary to previous research, the estimation problems in this thesis are solved in a rigorous Bayesian way. This will, given enough computing power, allow us to deal with more complex models (larger uncertainties) than was previously possible. Graphical modeling allows us to choose the most appropriate filtering algorithm given a certain model and to study the consequences on an algorithmic level of choosing a more or less complex model.

Future research that focusses more on active sensing will have to deal with decision making. This Chapter presented some hints to deal with active sensing under large uncertainties, which are complementary to the work of Lefebvre (2003), who considers the state estimates to be the true values and does not consider the related uncertainty (both with a unimodal or multimodal posterior density). Unfortunately, it is computationally unfeasible to apply optimization algorithms capable of dealing with uncertainty such as partially observable Markov decision processes (POMDPs). Even for the small uncertainties, computations time seems prohibitively high. Therefore, ad hoc problem dependent policies seem to be the only solution to the active sensing problem under large uncertainties.

Chapter 3

Monte Carlo methods

3.1 Introduction

This Chapter provides an overview of the Monte Carlo methods used in the remainder of this thesis, with focus on its applications in recursive Bayesian estimation. Algorithms from literature frequently used in this thesis (e.g. in BFL, Chapter 7) are in explicit form in the text.

This thesis mainly uses Monte Carlo methods to perform numerical integration.[1] Indeed, as described in Section 2.7, estimation is only one part of the job in order to autonomously perform tasks, and the full posterior PDF cannot be used in deterministic algorithms performing other subtasks. Typically, the quantities of interest for deterministic algorithms are the *moments* (e.g. the expected value or the variance), or the entropy of the posterior; another use of Monte Carlo methods occurs in marginalization (Section 2.4). All these operations involve integration.

A variety of methods for numerical integration exists. The most famous methods are probably quadrature methods, but the statistics community developed some well known approximations that yield good results in particular circumstances, such as *Laplace's method* (Kass and Raftery 1995) or *Variational Bayes* (Jaakkola and Jordan 1999; Jaakkola and Jordan 2000). Minka (2001), Chapter 2, provides an overview.

Monte Carlo methods (Metropolis and Ulam 1949) are a group of algorithms in which physical or mathematical problems are solved by using (pseudo-)Random Number Generators. The name "Monte Carlo" was chosen by Metropolis during the Manhattan Project of World War II, because of the similarity of statistical simulation to games of chance—and the capital of Monaco was a center for gambling and similar pursuits—. Monte Carlo

[1] Or the calculation of sums in the case of discrete variables.

methods were first used to perform simulations on the collision behavior of particles[2] during their transport within a material (to make predictions about how long it takes to collide).

A lot of books have been written about Monte Carlo methods, e.g. (Hammersley and Handscomb 1964; Rubinstein 1981; Kalos and Whitlock 1986; Liu 2001). This Chapter does not attempt to provide a complete overview of Monte Carlo methods. For a more general and cross-disciplinary review of Monte Carlo methods, see e.g. (Iba 2001).

This Chapter is organized as follows: Section 3.2 defines Monte Carlo methods more formally, and discusses some factors that influence convergence and performance. The remainder of the Chapter contains an overview of different Monte Carlo methods, with a focus on the methods used in this thesis. Figure 3.1 gives an overview of all discussed sampling methods.

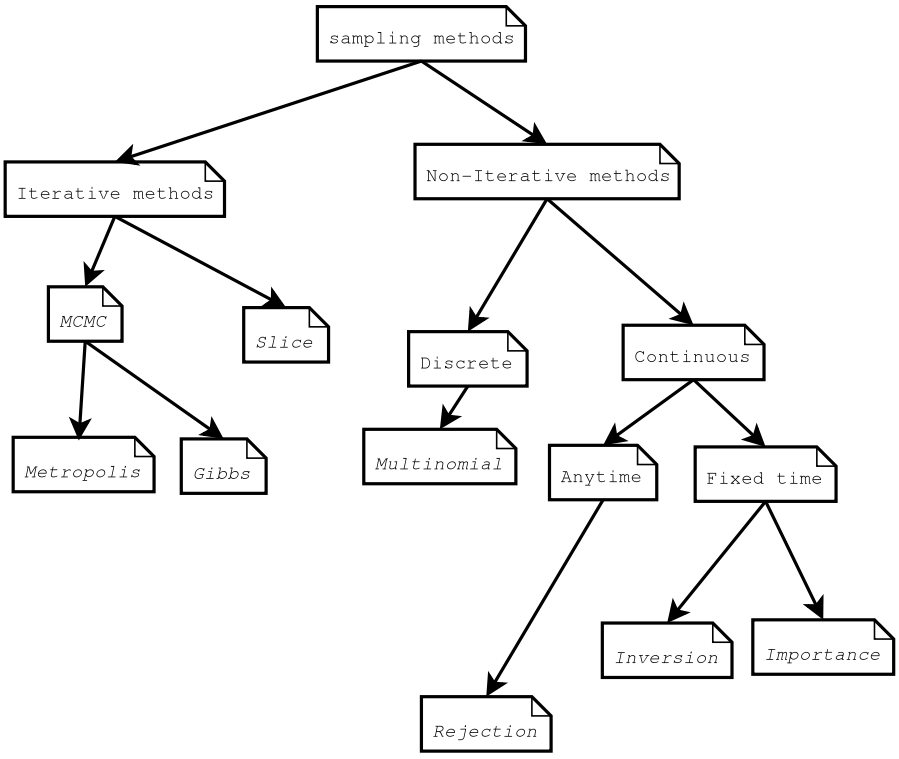


Figure 3.1: Overview of different MC methods.

[2]Therefore random samples are often denoted as *particles* in literature.

3.2 Monte Carlo methods

Monte Carlo techniques provide a number of ways to solve the sampling, estimation and marginalization problems:

- **Sampling.** Drawing *independent and identically distributed* (*i.i.d.*) samples[3] from a PDF (commonly referred to as "sampling *from* a certain PDF"). All sampling methods discussed in the remainder of this Chapter require the use of a (pseudo-)Random Number Generator (RNG). A lot of research has been done on the design/tests of pseudo-RNG algorithms; this thesis does not go into detail about the subject, for a detailed treatment on the matter, see e.g. (Devroye 1985; Marsaglia and Zaman 1993).

- **Estimation.** Estimating the value of a particular function of interest $h(\cdot)$ of an unknown variable $\boldsymbol{x}$:

$$I = E_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right] = \int h(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \tag{3.1}$$

  if $\boldsymbol{x}$ is continuous, or

$$I = E_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right] = \sum_i h(\boldsymbol{x}_i)p(\boldsymbol{x}_i), \tag{3.2}$$

  if $\boldsymbol{x}$ is discrete.[4] $h(\boldsymbol{x})$ denotes the function of interest. E.g. if $h(\boldsymbol{x}) = \boldsymbol{x}, I = E_{p(\boldsymbol{x})}\left[\boldsymbol{x}\right]$ represents the expected value of the PDF $p(x)$.

  If Monte Carlo methods are used in a Bayesian context to estimate the posterior characteristics, $p(\boldsymbol{x})$ denotes the posterior PDF.[5]

- **Marginalization.** Equation 2.9, the system update in case of filtering involving marginalization over $\boldsymbol{x}_{k-1}$, is a particular example of equation eq. (3.1). In this case, $p(\boldsymbol{x})$ is the prior PDF $P(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1})$.

Once we are able to sample from $p(\boldsymbol{x})$, (3.1) is approximated by:

$$I \approx \hat{I} = \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{x}^i) \qquad \boldsymbol{x}^i \sim p(\boldsymbol{x}), \tag{3.3}$$

[3] In stochastical terms, a sample is that part of a population which is actually observed.

[4] Or of course a mixture in case $\boldsymbol{x}$ is hybrid (i.e. partly continuous, partly discrete).

[5] Note that $\boldsymbol{x}$ in this Chapter denotes a particular value of interest, and no distinction is made between states and parameters. So $\boldsymbol{x}$ can represent $\boldsymbol{x}_k$, $\boldsymbol{x}_{1:k}$, $(\boldsymbol{x}_k, \boldsymbol{\theta})$, ... throughout the whole Chapter.

where $\boldsymbol{x}^i \sim p(\boldsymbol{x})$ denotes that $\boldsymbol{x}^i$ is a sample drawn from $p(\boldsymbol{x})$.
$\hat{I} \to I$ when $N \to \infty$. Indeed, let $\boldsymbol{x}^i \sim p(\boldsymbol{x})$ and define

$$F = \sum_{i=1}^{N} \lambda_n f_n(\boldsymbol{x}^i). \tag{3.4}$$

Then F is a random variable with expectation

$$E_{p(\boldsymbol{x})}[F] = E_{p(\boldsymbol{x})}\left[\sum_{i=1}^{N} \lambda_n f_n(\boldsymbol{x}^i)\right] = \sum_{i=1}^{N} \lambda_n E_{p(\boldsymbol{x})}\left[f_n(\boldsymbol{x}^i)\right] = \sum_{i=1}^{N} \lambda_n E_{p(\boldsymbol{x})}\left[f_n(\boldsymbol{x})\right].$$

Now suppose $\lambda_n = \dfrac{1}{N}$ and $f_n(\boldsymbol{x}) = h(\boldsymbol{x}) \quad \forall n$, then

$$E_{p(\boldsymbol{x})}[F] = \sum_{i=1}^{N} \frac{1}{N} E_{p(\boldsymbol{x})}[h(\boldsymbol{x})] = E_{p(\boldsymbol{x})}[h(\boldsymbol{x})] = I.$$

This means that, if N is large enough, $\hat{I}$ will converge to I.

3.2.1 Convergence of Monte Carlo methods

Starting from the *Central Limit Theorem* (CLT, asymptotically for $N \to \infty$), one can obtain expressions that indicate how good approximation (3.3) is. E.g. using the CLT:

$$\sqrt{N}\left(\hat{I} - I\right) \to \mathcal{N}\left(0, \sigma^2\right), \tag{3.5}$$

where $\sigma^2 = Var_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right]$. This means the convergence rate of Monte Carlo methods is $\mathcal{O}\left(1/\sqrt{N}\right)$, which is commonly referred to as "Monte Carlo methods don't suffer from the *curse of dimensionality*". Indeed, "traditional" grid-based methods for evaluating eq. (3.1) (such as *Riemann integration*) have a convergence rate of $\mathcal{O}\left(1/N\right)$ but require the evaluation of N^{n_x} points, where n_x denotes the dimension of $\boldsymbol{x}$ and N the number of grid points in one dimension.

Result (3.5) has to be interpreted with care though:

- $Var_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right]$ can be high over the region D to be integrated, resulting in slow convergence;
- Obtaining i.i.d. samples over D might not be trivial and time/resource consuming.

The first problem is independent of the sample method used, the second is not.

3.2.2 Performance of Monte Carlo methods

As eq. (3.5) shows, for a given and fixed number of samples, the performance/quality of the Monte Carlo method depends on

$$Var_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right] = E_{p(\boldsymbol{x})}\left[\left(h(\boldsymbol{x}) - E_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right]\right)^2\right] = E_{p(\boldsymbol{x})}\left[\left(h(\boldsymbol{x}) - I\right)^2\right],$$

and thus on the unknown quantity I.

Suppose that, instead of sampling from $p(\boldsymbol{x})$, we could sample from a density $\pi(\boldsymbol{x})$ proportional to $h(\boldsymbol{x})p(\boldsymbol{x})$, the resulting variance is zero and we obtain an exact estimate of x. This is useful if we are able to draw i.i.d. samples from $\pi(\boldsymbol{x})$. Although this scenario will never occur in real life problems,[6] the so-called *importance sampling* methods (Section 3.5) can speed up the convergence process by choosing an appropriate density π. This principle is often denoted as *variance reduction*. The performance of importance sampling methods is treated in detail in Section 3.5.3.

Several methods exist to reduce the Monte Carlo variance, such as *stratified sampling* (Kitagawa 1996), *control variates* and *antithetic variates* (see e.g. (Liu 2001)), and *quasi-Monte Carlo Methods* (Fearnhead 1998; Fearnhead). Unfortunately, these methods require careful application and are only useful in specific cases: e.g. stratified sampling approximates the proposal density by a piecewise constant and is useful for reducing Monte Carlo variation if the proposal can easily be approximated by such a piecewise constant. A particular useful application is found during the resampling stage of Particle Filters (Section 4.4.1), where samples have to be drawn from a uniform density in $[0, 1]$.

General guidelines to improve the performance of Monte Carlo methods do not exist. An efficient software implementation of a basic algorithm combined with lots of samples might sometimes be the fastest and most straightforward way to obtain good results.

Another technique worth mentioning in this scope is the principle of *Rao-Blackwellization* (Casell and Robert 1996). In a lot of applications, the stochastic variable $\boldsymbol{X}$ is composed of several parts $\boldsymbol{X}^i$. By exploiting conditional independence relations between those parts, it is sometimes possible to perform parts of the evaluation of eq. (3.1) analytically. A typical example is (quasi-)linear conditional relationship $P(\boldsymbol{X}^j|\boldsymbol{X}^i)$ with additive Gaussian uncertainty; in that case the conditional expectation can be modeled using an analytic filter such as the Kalman Filter. A recent application of this principle is the FastSLAM algorithm (Montemerlo, Thrun, Koller, and Wegbreit

[6] Indeed, it is already very hard to draw samples from most PDFs. Drawing samples from a PDF proportional to the product of is only possible in a very limited number of cases! E.g. drawing samples from a Gaussian $p(\boldsymbol{x}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is possible, but drawing samples from $xp(\boldsymbol{x})$ is already impossible.

2003) in mobile robotics. In this perspective, the construction of a graphical model often helps in choosing the most appropriate filtering algorithm (See Chapter 5).

The remainder of this Chapter describes several methods for drawing i.i.d. samples from certain distributions, that will be used in later Chapters of this thesis. For a more in depth review of Monte Carlo methods, we refer to (Gilks, Richardson, and Spiegelhalter 1996; Robert and Casella 1999; Iba 2001; Liu 2001).

3.3 Sampling from a discrete distribution

Sampling from a discrete distribution is an important problem, due to its frequent use in the resampling step of Particle Filters (Section 4.4.1). The basic algorithm is often called *multinomial sampling* (a multinomial distribution is an extension of the *binomial distribution* to the case where there are more than two classes into which an event can fall), and is shown in algorithm 3.1. Multinomial sampling takes $\mathcal{O}(NlogN)$ time to draw N samples. This would be the bottleneck for application during sequential Monte Carlo procedures. More efficient methods ($\mathcal{O}(N)$) exist and are commonly used in Particle Fil-

Algorithm 3.1 Multinomial Sampling. CDF represents the *Cumulative Distribution Function*. That is, since $\boldsymbol{x}$ is always discrete in this case: $CDF(\boldsymbol{x}_i) = \sum_{j=0}^{i} p(\boldsymbol{x}_j)$.

Construct the CDF of $p(\boldsymbol{x}_i)$: $CDF(\boldsymbol{x}_i)$.
Sample N samples $u_i \quad (1 < i <= N)$ from a uniform density $U[0,1]$
for $i = 1$ to N **do**
  $j = 0$
  while $u_i > CDF(\boldsymbol{x}_j)$ **do**
    $j++$
  end while
  Add $\boldsymbol{x}_j$ to sample list
end for

ters, e.g. using *ordered* uniform samples (Ripley 1987), p. 96 (algorithm 3.2) or methods based on arithmetic coding (MacKay 2003), Section 6.3.

3.4 Inversion sampling

If we transform a stochastic variable $\boldsymbol{X}$ into another one $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X})$, the invariance rule states that the probability mass in a certain part of the param-

Algorithm 3.2 Ordered resampling.

Construct the CDF of $p(\boldsymbol{x}_i)$: $CDF(\boldsymbol{x}_i)$.
Sample N samples $u_i \quad (1 < i <= N)$ from a uniform density $U[0,1]$
Take N^{th} root of u_N: $u_N = u_N^{1/N}$
for $i = N-1$ to 1 **do**
  Rescale sample: $u_i = u_i^{(1.0/i)} \times u_{i+1}$
end for
$j = 0$
for $i = 1$ to N **do**
  while $u_i > CDF(\boldsymbol{x}_j)$ **do**
    $j++$
  end while
  Add $\boldsymbol{x}_j$ to sample list
end for

eter space of the distribution must remain the same in both representations:

$$p(\boldsymbol{x})d\boldsymbol{x} = p(\boldsymbol{y})d\boldsymbol{y}, \tag{3.6}$$

and thus

$$p(\boldsymbol{y}) = \frac{p(\boldsymbol{x})}{\frac{d\boldsymbol{y}}{d\boldsymbol{x}}}.$$

Suppose we want to generate samples from a certain PDF $p(\boldsymbol{x})$. If we take the transformation function $\boldsymbol{y} = f(\boldsymbol{x})$ to be the cumulative distribution function (CDF) from $p(\boldsymbol{x})$, $p(\boldsymbol{y})$ will be a uniform distribution on the interval $[0,1]$. So, if $p(\boldsymbol{x})$ is an analytical function, and the inverse CDF f^{-1} exists and can be calculated, sampling is straightforward (algorithm 3.3). The obtained samples

Algorithm 3.3 Inversion sampling.

for $i = 1$ to N **do**
  Sample $\boldsymbol{u}^i$ from $U[0,1]$
  $\boldsymbol{x}^i = f^{-1}(\boldsymbol{u}^i)$ where $f(\boldsymbol{x}) = \int_{-\infty}^{x} p(\boldsymbol{x})d\boldsymbol{x}$
end for

$\boldsymbol{x}^i$ are *exact* samples from $p(\boldsymbol{x})$. This approach is illustrated in figure 3.2.

An important case of this method is the *Box-Muller* method, because it draws samples from a *univariate* normal distribution (see e.g. (Kalos and Whitlock 1986)).

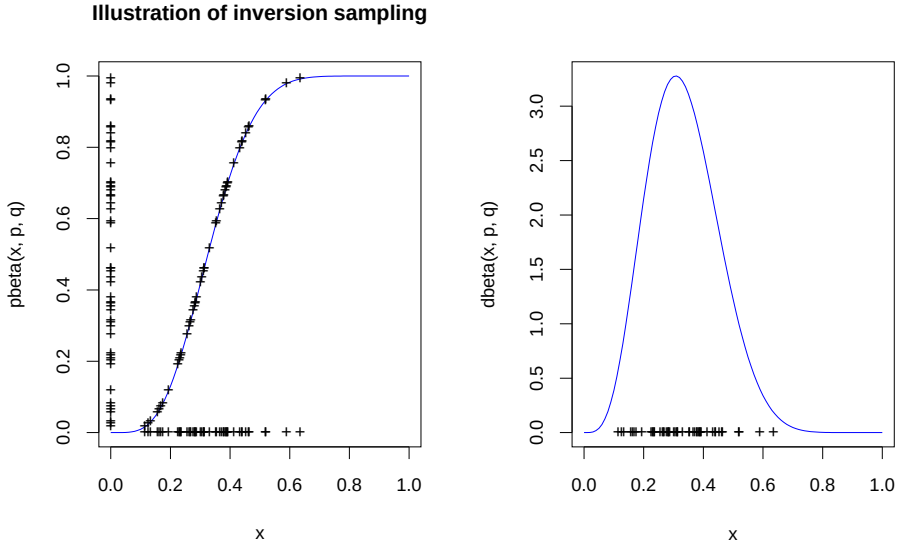


Figure 3.2: Inversion sampling: 50 samples from a uniform RNG (on the vertical axis), transformed through the inverse cumulative Beta distribution to samples from the Beta distribution on the horizontal axis. The right hand side plots these samples underneath the Beta distribution.

If u_1, u_2 are independent and uniformly distributed,

$$x_1 = \sqrt{-2 \log u_1} \cos(2\pi u_2),$$
$$x_2 = \sqrt{-2 \log u_1} \sin(2\pi u_2),$$

are independent samples from a standard normal distribution. These *univariate* standard normal samples can in turn be used to generate samples from a *multivariate* normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Suppose $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ (e.g. by a Choleski decomposition, which is always possible, since a covariance matrix should be positive semi-definite and symmetric), and $\boldsymbol{Z}$ is a vector where each element is an i.i.d. sample from a standard normal distribution. Then, $X = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{Z}$ is an i.i.d. sample from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, since $E[\boldsymbol{X} - \boldsymbol{\mu}] = \mathbf{0}$ and $E\left[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^T\right] = \boldsymbol{L}E[\boldsymbol{Z}\boldsymbol{Z}^T]\boldsymbol{L}^T = \boldsymbol{\Sigma}$. Since most system and measurement models in robotics represent uncertainty by adding Gaussian noise, this sampling algorithm is one of the most frequent operations when applying Particle Filters.

Variations on the inversion sampling method exist: the *approximate inversion sampling* method applies inversion sampling to a discretisation of the distribution we want to sample from.

3.5 Importance sampling

Importance sampling methods (Bernardo and Smith 1994) are used to obtain *approximate* samples from a *proposal* distribution when *exact* sampling methods are not applicable (Section 3.5.1). The *proposal distribution* is a PDF of which one can draw samples by using one of the exact sampling methods described in this Chapter such as inversion sampling. Section 3.5.2 describes how importance sampling methods are used to approximate eq. (3.3). Sections 3.5.3 and 3.5.4 deal with the choice of the proposal PDF in importance sampling and with the consequences on the variance of the estimates.

3.5.1 Obtaining approximate samples from a PDF

In many cases $p(\boldsymbol{x})$ is too complex to compute f^{-1} explicitly, so inversion sampling isn't possible. One approach to solve this is to approximate $p(\boldsymbol{x})$ by a function $q(\boldsymbol{x})$ to which *exact* sampling methods apply. $q(\boldsymbol{x})$ is often called the *proposal density* (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953)[7] or the *importance function* (Doucet 1998; Doucet 1997). Algorithm 3.4 shows how to obtain *approximate* samples from $p(\boldsymbol{x})$ with this technique. It is sometimes referred to as *Sampling Importance Resampling* (SIR) and

Algorithm 3.4 Generating samples using Importance Sampling.

Require: $M >> N$
  for $j = 1$ to M **do**
    Sample $\tilde{\boldsymbol{x}}^j \sim q(\boldsymbol{x})$ {e.g. with the inversion technique}
    $w_j = \dfrac{p(\tilde{\boldsymbol{x}}^j)}{q(\tilde{\boldsymbol{x}}^j)}$
  end for
  for $i = 1$ to N **do**
    Sample $\boldsymbol{x}^i \sim (\tilde{\boldsymbol{x}}^j, w_j) \quad 1 < j < M$ {Discrete distribution}
  end for

was originally described by Rubin (1988) to perform inference in a Bayesian context. Rubin drew samples from the prior distribution, assigned a weight to each of them according to their likelihood. I.i.d. samples from the posterior distribution are obtained by resampling from the latter discrete set.[8] This approach is illustrated in figure 3.3.

[7] This term is also used in MCMC methods where it has a different meaning, see Section 3.7.

[8] Liu (2001) denotes such as set of weighted samples a *properly weighted* sample set.

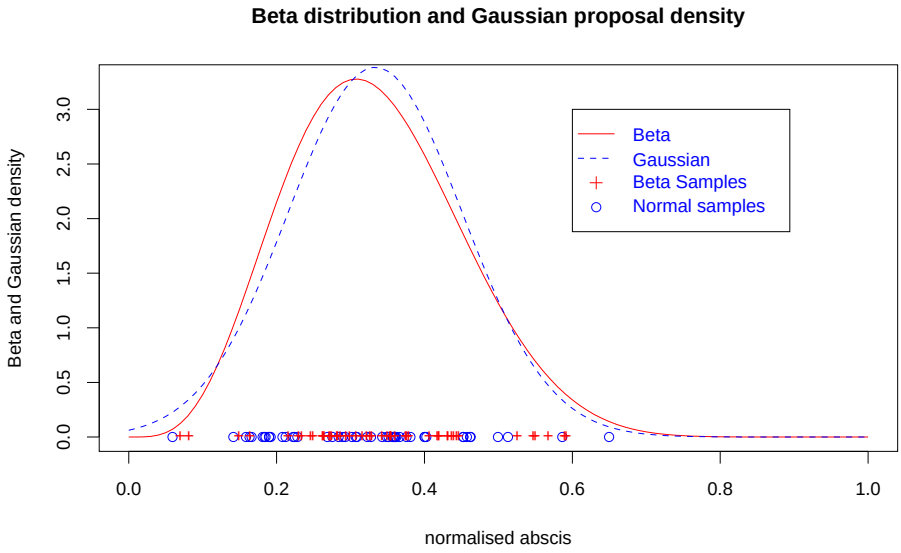


Figure 3.3: Illustration of importance sampling. Drawing samples from a Beta distribution via a Gaussian with the same mean and covariance.

3.5.2 Monte Carlo Integration using Importance Sampling

Importance sampling is also used to provide an estimate of eq. (3.1) if i.i.d. samples from $p(\boldsymbol{x})$ are hard to obtain, or as described in Section 3.2.2, it can be important to speed up convergence in case the variance $Var_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$ is high. If we can sample approximately from a proposal distribution $q(\boldsymbol{x}) \approx p(\boldsymbol{x})$ and define

$$w(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}, \tag{3.7}$$

we can rewrite eq. (3.1) as

$$I = E_{p(\boldsymbol{x})}[h(\boldsymbol{x})] = \int h(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \int h(\boldsymbol{x})w(\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}. \tag{3.8}$$

This leads to the following (biased) approximation of (3.1) using samples $\boldsymbol{x}^i$ drawn independently from a distribution $q(\boldsymbol{x})$:[9]

$$\hat{I} = \frac{\sum_{i=1}^{N} h(\boldsymbol{x}_i)w_i}{\sum_{i=1}^{N} w_i}, \tag{3.9}$$

where w_i denotes $w(\boldsymbol{x}^i)$.

As mentioned in Section 3.2.2, choosing a proposal density similar to $h(\boldsymbol{x})p(\boldsymbol{x})$ instead of just $p(\boldsymbol{x})$, would increase the convergence speed (still $\mathcal{O}\left(1/\sqrt{N}\right)$ though). In practice, this is most often very hard to achieve. Algorithm 3.5 recalls both cases.

3.5.3 Performance of Importance Sampling methods

As stated in Section 3.2.2, the "absolute" performance of Monte Carlo methods depends on $Var_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$ (eq. (3.5)). Since sampling from the product $h(\boldsymbol{x})p(\boldsymbol{x})$ is quasi-impossible for real-life problems (and gets more difficult as the dimension of the random variables increases), most importance sampling methods focus on providing an approximation of $p(\boldsymbol{x})$. This has the extra "advantage" that only one importance sampling density has to be chosen that can be used for evaluating different functions of interest $h(\boldsymbol{x})$ (See algorithm 3.5).

The resulting Monte Carlo variance of an estimator is the sum of the problem-specific part, due to $h(\boldsymbol{x})$, and the proposal-specific part, due to the

[9] The denominator of this equation stems from the fact that the weights are mostly only proportional to the ratio $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$. E.g. in a Bayesian context, the denominator represents an approximation of the *evidence* (See eq. (2.6)).

Algorithm 3.5 Integral estimation using Importance Sampling.

for $i = 1$ to N **do**
Sample $\boldsymbol{x}^i \sim q(\boldsymbol{x})$ {where $q(\boldsymbol{x}) \approx h(\boldsymbol{x})p(\boldsymbol{x})$ or $q(\boldsymbol{x}) \approx p(\boldsymbol{x})$. Note that the former requires a *different* proposal density for each function of interest $h(\boldsymbol{x})$.}
$w_i = \dfrac{p(\boldsymbol{x}^i)h(\boldsymbol{x}^i)}{q(\boldsymbol{x}^i)}$ or $w_i = \dfrac{p(\boldsymbol{x}^i)}{q(\boldsymbol{x}^i)}$
end for

$$I \approx \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} x_i \text{ or } I \approx \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} h(\boldsymbol{x}_i) w_i$$

difference between $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$. Comparing the performance of different importance sampling methods should only compare the second part.

The performance of importance sampling methods is discussed by (Ripley 1987; Geweke 1989; Liu 1996; Neal 1998) in a similar way. Based on the delta method, they calculate an approximation to the estimator's variance:

$$Var\left[\hat{I}\right] \approx \frac{Var_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right]\left(1 + Var_{q(\boldsymbol{x})}\left[w(\boldsymbol{x})\right]\right)}{N}. \tag{3.10}$$

This Taylor-series approximation is only valid if the covariance between the weights and $h(\boldsymbol{x})$ is ignored. The $h(\boldsymbol{x})$-independent part of this equation gives rise to the definition of the *effective sample size* (ESS):

$$ESS = \frac{N}{1 + Var_{q(\boldsymbol{x})}\left[w(\boldsymbol{x})\right]}. \tag{3.11}$$

If we could draw N exact samples from $p(\boldsymbol{x})$, the estimators variance would be

$$Var\left[\hat{I}\right] = \frac{Var_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right]}{N}. \tag{3.12}$$

Therefore, the effective sample size gives the number of samples that are necessary to achieve the same variance if those samples were drawn from the true target density $p(\boldsymbol{x})$ instead of the proposal. The second term in the denominator is unknown but can be estimated by the sample variance, resulting in

$$\frac{N}{1 + \sum_{i=1}^{N} \left(\tilde{w}^i\right)^2}, \tag{3.13}$$

where $\tilde{w}$ denotes a normalized weight (since the target distribution is only known up to a constant in most applications). Note that the sample variance can also be a bad estimate of $Var_{q(\boldsymbol{x})}\left[w(\boldsymbol{x})\right]$, e.g. due to the choice of a bad

proposal density (see the next Section). This can lead to an ESS that is a very bad estimate of the true performance (Neal 1998).

Although not always reliable due to the above assumptions, the ESS is the most frequent criterion to assess the performance of an importance sampling algorithm, e.g. in Particle Filters with a *dynamic* resampling schedule (Section 4.4.1).

Fearnhead (1998) doesn't make the assumption about the covariance term being zero and calculates the *true* Monte Carlo variance (also based on the delta method) as

$$ESS_{true} = \frac{N}{E_{q(\boldsymbol{x})}\left[(h(\boldsymbol{x}) - I)^2 w(\boldsymbol{x})^2\right]}, \tag{3.14}$$

with an error term $\mathcal{O}\left(1/N^{3/2}\right)$ (due to the Taylor series approximation). Unfortunately, since this equation contains the unknown quantity I, it is impossible to derive a practical implementation of this. Fearnhead gives some examples of cases where the ESS is infinitely wrong ($ESS/ESS_{true} = 0$). Another way to estimate the ESS, as proposed by (Carpenter, Clifford, and Fearnhead 1999b) is to run several importance sampling algorithms in parallel, each started with a different random seed and to use these results to calculate a Monte Carlo estimate of the ESS. Unfortunately, this method requires much more resources, and is therefore not used in sequential Monte Carlo methods.

3.5.4 Choosing the proposal density

From the previous section, it is obvious that the importance density $q(\cdot)$ should be a reasonable approximation of $p(\cdot)$ (or $p(\cdot)h(\cdot)$): the further $p(\cdot)$ and $q(\cdot)$ are apart (e.g. in terms of Kullback-Leibler pseudo-distance (Kullback and Leibler 1951)), the bigger the variance of the weights will be and the worse the ESS.

A second important point is that $q(\cdot)$ should have at least as heavy tails as $p(\cdot)$, to avoid numerical degeneracy of the weight factor (i.e. the ratio $p(\cdot)/q(\cdot)$ gets too big in the tails, $Var_{q(\boldsymbol{x})}\left[w(\boldsymbol{x})\right]$ increases and the ESS decreases). This is confirmed by Geweke (1989) (theorem 3), who proves that the optimal density is $p(x)|h(x) - I|$,[10] having heavier tails than $p(\cdot)$.

3.5.5 Applications

Importance Sampling is a *fixed period* algorithm where the quality of the approximation depends on the number of samples used and the chosen proposal density. Due to its fixed period properties, it is very commonly used for

[10] Note that I is unknown!

sequential estimation and it is the basic algorithm for most Particle Filter algorithms (Chapter 4). In other cases, where there are no realtime fixed-period constraints or when *anytime algorithms* (not guaranteeing any convergence, but providing an answer at any given time) apply, or where *exact* samples are required, other algorithms can be used. One of them, *Rejection Sampling* is described in the next section.

3.6 Rejection sampling

Rejection sampling (e.g. (Hammersley and Handscomb 1964)) is an *exact* sampling algorithm with *anytime* properties. It uses a proposal density $q(\boldsymbol{x})$ such that

$$cq(\boldsymbol{x}) > p(\boldsymbol{x}), \quad \forall \boldsymbol{x}. \tag{3.15}$$

The procedure is as follows: generate samples from q. For each sample $\boldsymbol{x}^i$, generate a value uniformly drawn from the interval $[0, cq(\boldsymbol{x}^i)]$. If the generated value is smaller than $p(\boldsymbol{x}^i)$, the sample is accepted, else the sample is rejected. This approach is illustrated by algorithm 3.6 and figure 3.4. Re-

Algorithm 3.6 Rejection Sampling algorithm.

$j = 1, i = 1$
repeat
  Sample $\tilde{\boldsymbol{x}}^j \sim q(\boldsymbol{x})$
  Sample $\boldsymbol{u}^j$ from $U[0, cq(\tilde{\boldsymbol{x}}^j)]$
  if $\boldsymbol{u}^j < p(\tilde{\boldsymbol{x}}^j)$ **then**
    $\boldsymbol{x}^i = \tilde{\boldsymbol{x}}^j$ {Accepted}
    $i++$
  end if
  $j++$
until $i = N$
Acceptance Rate $= \dfrac{N}{j}$

jection sampling is only interesting if the number of rejections is small: the Acceptance Rate (as calculated in algorithm 3.6) should be as close to 1 as possible. Therefore the proposal density q has to be similar to $p(\boldsymbol{x})$). Just as with importance sampling, this becomes more difficult for high dimensional problems. Another problem with rejection sampling is the calculation of c.

In Particle Filters, *Rejection Control* methods are sometimes combined with Importance Sampling (Liu 2001; Liu and Chen 1998) (adjusting proposal density in light of current importance weights). This is more thoroughly discussed in Section 4.4.3.

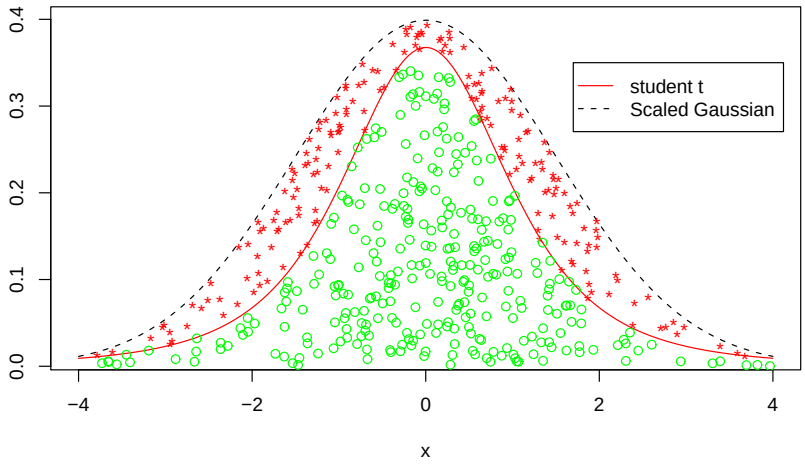


FIGURE 3.4: Rejection sampling. The circles (∘) denote accepted samples, the stars (⋆) denote rejected samples.

3.7 Markov Chain Monte Carlo (MCMC) methods

Importance and rejection sampling only yield acceptable results if the proposal density $q(\boldsymbol{x})$ approximates $p(\boldsymbol{x})$ fairly well. For high dimensional problems, this is often utopian. Markov Chain MC methods (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Gelfland and Smith 1990; Smith and Gelfland 1992; Robert and Casella 1999) are *iterative* methods that use Markov Chains to sample from PDFs and don't suffer from this drawback. On the downside, next to the fact that they are hard to use in a sequential context, they provide correlated samples and it might take a large number of transition steps of the Markov Chain in order to explore the whole state space.

This section first discusses the general principle of MCMC sampling—the Metropolis–Hastings algorithm (Hastings 1970; Chib and Greenberg 1995)—, and then focusses on some particular implementations. The described algorithms and more variations are more thoroughly discussed in (Neal 1993; MacKay 2003; Gilks, Richardson, and Spiegelhalter 1996; Casella and George 1992). MCMC methods are computationally too expensive to be used for

recursive sequential estimation, but a MCMC step is sometimes performed to avoid degeneracy, see Section 4.4.2. Unfortunately, this is not possible in a fixed period.

3.7.1 The Metropolis–Hastings algorithm

This algorithm is often referred to as the $M(RT)^2$ algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953), although its most general formulation is due to Hastings (1970). Therefore, it is called the Metropolis–Hastings algorithm. It provides samples from $p(\boldsymbol{x})$ by using a Markov Chain:

- Choose a proposal density $q(\boldsymbol{x}, \boldsymbol{x}^{(t)})$, (that can but *need* not be) depending on the current sample $\boldsymbol{x}^{(t)}$. Contrary to the previous sample methods, the proposal density doesn't have to be similar to $p(\boldsymbol{x})$. It can be any density from which we can draw samples. We assume we can evaluate $p(\boldsymbol{x})$ for all $\boldsymbol{x}$.
  Choose an initial state $\boldsymbol{x}^0$ of the Markov Chain.
- At every time step t, generate a new state $\tilde{\boldsymbol{x}}$ from $q(\boldsymbol{x}, \boldsymbol{x}^{(t)})$. To decide if this new state will be accepted, compute

$$a = \frac{p(\tilde{\boldsymbol{x}})}{p(\boldsymbol{x}^{(t)})} \frac{q(\boldsymbol{x}^{(t)}, \tilde{\boldsymbol{x}})}{q(\tilde{\boldsymbol{x}}, \boldsymbol{x}^{(t)})}. \tag{3.16}$$

  If $a \geq 1$, the new state $\tilde{\boldsymbol{x}}$ is accepted and $\boldsymbol{x}^{(t+1)} = \tilde{\boldsymbol{x}}$, else the new state is accepted with probability a (this means: sample a random uniform variable $\boldsymbol{u}^i$, if $a \geq \boldsymbol{u}^i$, then $\boldsymbol{x}^{(t+1)} = \tilde{\boldsymbol{x}}$, else $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$).

This approach is illustrated in figure 3.5. Asymptotically, the samples generated from this Markov Chain are samples from $p(\boldsymbol{x})$. Note that the generated samples are not i.i.d. drawn from $p(\boldsymbol{x})$, but correlated through the proposal density.

The following paragraphs show why, asymptotically, these samples are samples from $p(\boldsymbol{x})$, and discuss the consequences of using a Markov Chain and the influence of parameter choices on the convergence speed of the algorithm.

Convergence of the Markov Chain

Using Markov Chain theory, we can show that the Metropolis–Hastings algorithm asymptotically generates samples from $p(\boldsymbol{x})$. A (continuous) Markov Chain can be specified by an *initial PDF* $f^{(0)}(\boldsymbol{x})$ and a *transition PDF* or *transition kernel* $T(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = P(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. The PDF describing the state at the $(t+1)^{th}$ iteration of the Markov Chain, $f^{(t+1)}(\boldsymbol{x})$, is

$$f^{(t+1)}(\tilde{\boldsymbol{x}}) = \int T(\tilde{\boldsymbol{x}}, \boldsymbol{x}) f^{(t)}(\boldsymbol{x}) d\boldsymbol{x}.$$

A Markov Chain is *irreducible* if we can get from any state $\boldsymbol{x}$ into another state $\boldsymbol{y}$ within a *finite* amount of time. A distribution function $i(\boldsymbol{x})$ is called the *stationary* or *invariant distribution* from a Markov Chain with transition kernel $T(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ if

$$i(\tilde{\boldsymbol{x}}) = \int T(\tilde{\boldsymbol{x}}, \boldsymbol{x}) i(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.17}$$

An irreducible Markov Chain is called *aperiodic/acyclic* if there isn't any distribution function which allows something of the form

$$f(\tilde{\boldsymbol{x}}) = \int \cdots \int T(\tilde{\boldsymbol{x}}, \dots) \dots T(\dots, \boldsymbol{x}) f(\boldsymbol{x}) d \dots d\boldsymbol{x}, \tag{3.18}$$

where the dots denote a *finite* number of transitions! A Markov Chain which is both aperiodic and irreducible is said to be *ergodic*, and an ergodic Markov Chain is said to be time reversible if it satisfies the *detailed balance* property:

$$T(\boldsymbol{x}^a, \boldsymbol{x}^b) f(\boldsymbol{x}^b) = T(\boldsymbol{x}^b, \boldsymbol{x}^a) f(\boldsymbol{x}^a). \tag{3.19}$$

This property implies the invariance of the distribution $f(\boldsymbol{x})$ under the Markov Chain transition kernel $T(\tilde{\boldsymbol{x}}, \boldsymbol{x})$: Indeed, combine eq. (3.19) with the fact that

$$\int T(\boldsymbol{x}^a, \boldsymbol{x}^b) d\boldsymbol{x}^a = 1.$$

This yields

$$\begin{aligned} \int T(\boldsymbol{x}^a, \boldsymbol{x}^b) f(\boldsymbol{x}^b) d\boldsymbol{x}^a = & \int T(\boldsymbol{x}^b, \boldsymbol{x}^a) f(\boldsymbol{x}^a) d\boldsymbol{x}^a \\ f(\boldsymbol{x}^b) = & \int T(\boldsymbol{x}^b, \boldsymbol{x}^a) f(\boldsymbol{x}^a) d\boldsymbol{x}^a. \end{aligned}$$

It can also be proven that any ergodic Markov Chain that satisfies the detailed balance equation (3.19), will eventually converge to the invariant distribution of that chain $i(\boldsymbol{x})$, irrespective of the initial distribution $f^0(\boldsymbol{x})$. To prove that the Metropolis–Hastings algorithm does provide samples of $p(\boldsymbol{x})$, the latter has to be the invariant distribution for the Markov Chain with transition kernel defined by the MCMC algorithm. Using eq. (3.16), the transition kernel of the MCMC is

$$T(\boldsymbol{x}, \boldsymbol{x}^{(t)}) = q(\boldsymbol{x}, \boldsymbol{x}^{(t)}) \times a(\boldsymbol{x}, \boldsymbol{x}^{(t)}) + I(\boldsymbol{x} = \boldsymbol{x}^t) \left[1 - \int q(\boldsymbol{y} \mid \boldsymbol{x}^t) a(\boldsymbol{y}, \boldsymbol{x}^{(t)}) d\boldsymbol{y} \right], \tag{3.20}$$

where $I(\cdot)$ denotes the indicator function (taking the value 1 if its argument is true, and 0 otherwise). The probability of arriving in a state $\boldsymbol{x} \neq \boldsymbol{x}^t$ is just the first term of equation (3.20). The probability of staying in $\boldsymbol{x}^t$, on the

other hand, consists of two contributions: Either $\boldsymbol{x}^t$ was generated from the proposal density q and accepted, or another state was generated and rejected, since the integral "sums" over all possible rejections! To satisfy the detailed balance property if $\boldsymbol{x} \neq \boldsymbol{x}^{(t)}$ (the other case is trivial)

$$\begin{aligned} T(\boldsymbol{x}, \boldsymbol{x}^{(t)})p(\boldsymbol{x}^{(t)}) &= T(\boldsymbol{x}^{(t)}, \boldsymbol{x})p(\boldsymbol{x}) \\ q(\boldsymbol{x}, \boldsymbol{x}^{(t)})a(\boldsymbol{x}, \boldsymbol{x}^{(t)})p(\boldsymbol{x}^{(t)}) &= q(\boldsymbol{x}^{(t)}, \boldsymbol{x})a(\boldsymbol{x}^{(t)}, \boldsymbol{x})p(\boldsymbol{x}) \\ \frac{a(\boldsymbol{x}, \boldsymbol{x}^{(t)})}{a(\boldsymbol{x}^{(t)}, \boldsymbol{x})} &= \frac{q(\boldsymbol{x}^{(t)}, \boldsymbol{x})p(\boldsymbol{x})}{q(\boldsymbol{x}, \boldsymbol{x}^{(t)})p(\boldsymbol{x}^{(t)})}, \end{aligned}$$

the definition in eq. (3.16) suffices.

Efficiency considerations

Run length and Burn-in period The generated samples are only *asymptotically* samples from $p(\boldsymbol{x})$. This means a number of samples in the beginning of the algorithm (called the *burn-in period*, see e.g. (Gilks, Richardson, and Spiegelhalter 1996)) should be thrown away. Since the generated samples are also correlated through the proposal, the Markov Chain has to run long enough to explore the whole state space. So, eq. (3.1) is typically approximated by

$$E_{p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right] \approx \frac{1}{n-m} \sum_{i=m+1}^{n} f(\boldsymbol{x}^i), \tag{3.21}$$

where m denotes the burn-in period and the run length n should be large enough in order to assure the required precision and the exploration of the *whole* state space.
Several *convergence diagnostics* exist for determining both m and n (Gilks, Richardson, and Spiegelhalter 1996). The total number of samples n strongly depends on the ratio

$$\frac{\text{typical step size of Markov Chain}}{\text{representative length of State Space}}$$

of the algorithm (sometimes also called *convergence ratio*, although this term can be misleading).
This typical step size of the Markov Chain ϵ is related to the choice of the proposal density $q(\cdot)$ (e.g. $\epsilon = Var\left[q(\cdot)\right]$). If ϵ is too large, a lot of rejections will happen, if it's too small, it will take too long to explore the whole state space. To explore the whole state space efficiently (some authors use the term *well mixing* proposal density), the step size should be of the same order of magnitude as the smallest length scale of $p(\boldsymbol{x})$ (e.g. in the case $p(\boldsymbol{x})$ is a multimodal Gaussian, the step size has to be the of same order of magnitude

of the standard deviation from the Gaussian with the smallest covariance). In a sequential Monte Carlo context (e.g. when a MCMC step is added to a Particle Filter, see Section 4.4.2), typically one uses a Gaussian proposal with the current sample as mean and the sample variance as variance.

One way to determine the *stopping time*, given a required precision, is by tracking the Monte Carlo variance of the estimate in equation (3.21), but this can be misleading due to the dependence between the different samples. The most obvious method is starting several chains in parallel, and compare the different estimates. Again, the latter is far more time-consuming.

Independence Although the fact that samples are correlated hardly constitutes a problem in many practical cases for evaluation of the quantities of interest such as $E_{p(\boldsymbol{x})}[f(\boldsymbol{x})]$, a way to avoid dependence is to store only every lth value of the Markov Chain (often called *thinning*) and/or by starting different chains in parallel.

3.7.2 MCMC variants

The following paragraphs briefly describe some MCMC variants often found in literature.

Metropolis sampling Metropolis sampling (Metropolis et al. 1953) is a variant of Metropolis–Hasting sampling that supposes that the proposal density is symmetric around the current state.

The independence sampler The independence sampler is an implementation of the Metropolis–Hastings algorithm in which the proposal distribution is independent of the current state. This approach only works well if the proposal distribution is a good approximation of p (and heavier tailed to avoid getting stuck in the tails).

Single component Metropolis–Hastings For some multivariate densities, it can be very difficult to come up with an appropriate proposal density that explores the whole state space fast enough. Therefore, it is often easier to divide the state space vector $\boldsymbol{x}$ into a number of components:

$$\boldsymbol{x} = \left[\boldsymbol{x}_{.1}\boldsymbol{x}_{.2}\cdots\boldsymbol{x}_{.n}\right]^T,$$

where $\boldsymbol{x}_{.i}$ denotes the i-th component of $\boldsymbol{x}$. We can then update those components one by one. One can prove that this doesn't affect the invariant distribution of the Markov Chain.

Gibbs sampling Gibbs sampling (Geman and Geman 1984; Casella and George 1992) is a special case of the previous method. It's a Metropolis–Hastings algorithm, where the proposal distributions are the conditional distributions of the joint density $p(\boldsymbol{x})$. Gibbs sampling is an MCMC variant where the proposal is always accepted.
Gibbs sampling is probably the most popular form of MCMC sampling since it can easily be applied to *non-sequential inference problems*, if the problem exhibits *conditional conjugacy* properties (See Section 2.5.2). This is often the case for e.g. medical diagnosis problems with a lot of univariate variables, but hardly applicable to the 6-D localisation problems this thesis deals with.

3.7.3 Conclusions

Markov Chain Monte Carlo methods don't require the proposal distribution to be similar to the target distribution. However, drawbacks of MCMC methods are the fact that they are unsuited for use in online and recursive estimation due to their iterative nature. Therefore, they are typically not used for online estimation problems. Their samples are correlated and, in some cases, it is hard to choose some parameters in order to be able to explore the whole state space efficiently.[11]

3.8 Overview and Conclusions

This Chapter provided an overview of Monte Carlo methods that are used in the sequential Monte Carlo methods applied in this thesis. The main advantage of Monte Carlo methods is the fact that they don't suffer from the curse of dimensionality, which make them well suited to solve numerical integration problems in high dimensions. However, despite their theoretical $\mathcal{O}\left(1/\sqrt{N}\right)$ convergence, their application to complex problems is not straightforward, since the resulting covariance of their estimates can be very high. Furthermore, the number of samples necessary to achieve a certain precision is only determined by trial and error.

Importance sampling methods are the basic ingredient of the sequential Monte Carlo methods in Chapter 4. The multinomial sampling algorithm is the basis for stochastical resampling algorithms. Both rejection sampling methods and MCMC algorithms are used to avoid degeneracy problems with Particle Filters, although they are typically unsuited for online estimation problems.

[11] This Chapter does not discuss more advanced MCMC algorithms and methods, such as Hybrid Monte Carlo methods, simulated annealing, dynamical Monte Carlo, Stepping stones, ... See e.g (Neal 1998; Neal 2003; MacKay 2003) for an overview.

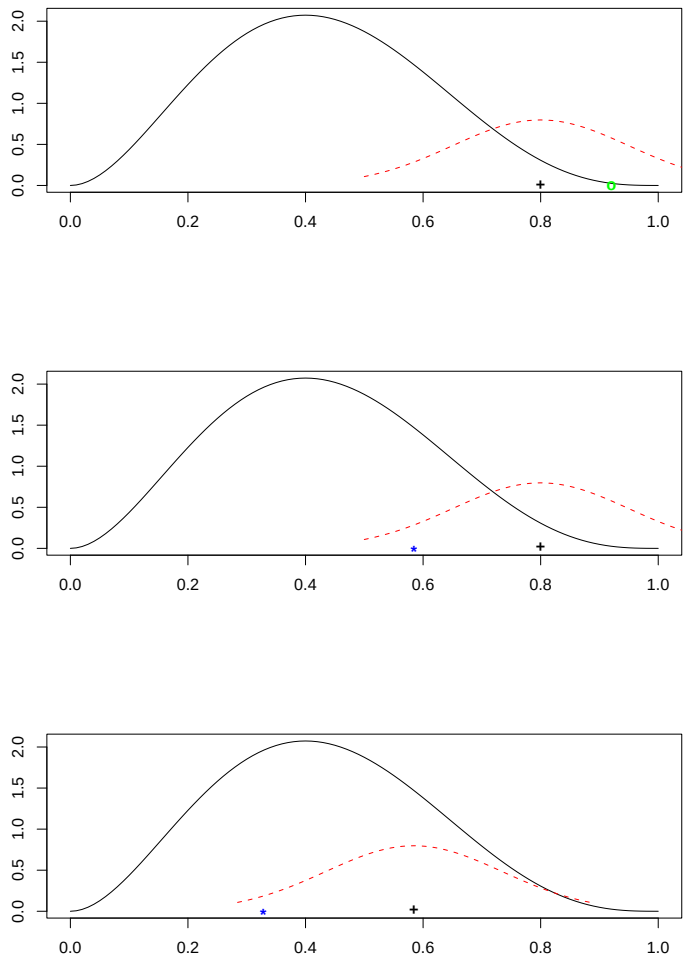


FIGURE 3.5: Demonstration of MCMC for a Beta distribution with a Gaussian proposal density. The Beta target density is the solid curve, the Gaussian proposal (centered around the current sample) is dotted, + denotes the current sample, ∘ denotes a rejected sample and ∗ and accepted sample.

Chapter 4

Sequential Monte Carlo methods

4.1 Introduction

Eq. (2.6), describing how to perform *Bayesian recursive inference*, does not specify anything about the *representation* of the PDF involved in the inference step. Depending on this representation, and taking into account assumptions about the models and/or priors, an enormous number of algorithms is available for the researcher. *Sequential Monte Carlo methods* (Doucet, de Freitas, and Gordon 2001) represent the posterior as a number of samples and allow to perform *recursive* Bayesian inference (i.e. the implementation of (2.6)). To that end, they apply one[1] of the Monte Carlo algorithms described in the previous Chapter in a recursive way (i.e. sequentially).

An algorithm of the above type was first *explicitly* proposed in (Handschin and Mayne 1969), but only became popular after being "reinvented" by (Gordon, Salmond, and Smith 1993). Since then, specific versions of sequential Monte Carlo methods appeared in research literature with various names: the *Condensation algorithm* (Isard and Blake 1998), the *Monte Carlo Filter* (Kitagawa 1993; Kitagawa 1996), *Survival of the fittest*, the *bootstrap filter* (Gordon, Salmond, and Smith 1993; Gordon, Salmond, and Ewing 1995), *interacting particle systems* (Crisan D. and Lyons 1999), *Sequential Importance Sampling*—with or without resampling (SIS/SIR) (Rubin 1988)—, ... The term *Particle Filter* is also often used as a synonym for the family of sequential Monte Carlo methods.

[1] Therefore, sequential Monte Carlo methods refer to a whole *family* of algorithms for performing recursive inference.

This Chapter first provides an introduction to one basic Particle Filter algorithm (The *Sequential Importance Sampling* (SIS) algorithm, Section 4.2), based on the principle of recursive Importance Sampling. This algorithm leaves the choice of the proposal density open. Section 4.3 discusses the choice of this proposal density in more detail. Due to the fact that the posterior density is represented by a *finite* number of particles, the basic algorithm suffers from numerical problems. Therefore, Section 4.4 provides an overview of some state of the art techniques to deal with this *impoverishment* phenomenon. Section 4.5 provides some insight in the theoretical domain of convergence results of Particle Filters and the related issue of the choice of the number of particles. Section 4.7 provides some pointers to soft– and hardware implementations. This Chapter does not compare the Sequential Monte Carlo methods with respect to other algorithms for recursive inference. This is discussed in Chapter 5.

For a good introduction to Particle Filters, see (Arulampalam, Maskell, Gordon, and Clapp 2002; Doucet, de Freitas, and Gordon 2001; Doucet 1998; Liu and Chen 1998). The Sequential Monte Carlo methods homepage (Doucet, de Freitas, and Punskaya) is a good starting point.

4.2 The Sequential Importance Sampling algorithm

The *Sequential Importance Sampling* (SIS) algorithm is a basic Particle Filtering algorithm that applies the principle of importance sampling recursively. A Particle Filter (PF) represents the posterior at each timestep k by a weighted set of samples, just as the importance sampling algorithm does this for the non-recursive case (Section 3.5).[2]

To apply importance sampling to represent the joint posterior $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$, one needs a proposal density

$$q(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}), \tag{4.1}$$

and the corresponding weights,

$$w_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}) = \frac{P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})}{q(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})}. \tag{4.2}$$

[2] In theory, the other sampling methods described in Chapter 3 could also be applied in a recursive way, however practically this is often infeasible. Hürzeler and Künsch (1995) describe a Particle Filter algorithm based on a sequential version of the rejection sampling algorithm as described in Section 3.6. However, the amount of computation to proceed from timestep k to $k+1$ of this algorithm is not fixed.

Since the goal is to obtain a recursive algorithm, the proposal density should, as the posterior, also be constructed in a recursive way using the product rule:

$$q(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) \triangleq q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k}) q(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}). \tag{4.3}$$

If we use the recursive factorization of the posterior from eq. (2.6) for the numerator,

$$P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}),$$

eq. (4.3) for the denominator and omit the normalization factor $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$, eq. (4.2) becomes

$$\begin{aligned} w_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}) &\sim \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1})}{q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k}) q(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1})} \\ &= \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k})} w_{k-1}(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}). \end{aligned} \tag{4.4}$$

The fraction in the latter equation is often referred to as the (unnormalized)[3] *incremental weight* $\hat{w}_k$. In the case of filtering, the incremental part of the proposal density should also be "Markovian":

$$q(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) \triangleq q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k) q(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}),$$

and the weight update becomes

$$\begin{aligned} w_k(\boldsymbol{x}_k, \boldsymbol{\theta}) &\sim \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k)} w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) \\ &= \hat{w}_k w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}), \end{aligned} \tag{4.5}$$

where $\hat{w}_k$ is the incremental weight at timestep k.

To cope with the normalization factor $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$, the weights are rescaled at each timestep in order to sum up to one:

$$\tilde{w}_k^i(\boldsymbol{x}_k, \boldsymbol{\theta}) = \frac{w_k^i(\boldsymbol{x}_k, \boldsymbol{\theta})}{\sum_{i=1}^N w_k^i(\boldsymbol{x}_k, \boldsymbol{\theta})}, \tag{4.6}$$

where N denotes the number of samples, and w_k^i denotes the weight evaluated at a certain particle value $(\boldsymbol{x}_k^i, \boldsymbol{\theta}^i)$.

The resulting set $\{(\boldsymbol{x}_{1:k}^i, \boldsymbol{\theta}^i), \tilde{w}^i\}$ (or $\{(\boldsymbol{x}_k^i, \boldsymbol{\theta}^i), \tilde{w}^i\}$ in the case of filtering) is used to estimate the properties of the posterior distribution as described in Section 3.5.2.

The resulting SIS algorithm is shown in algorithm 4.1.

[3]Unnormalized since we omitted $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$ from the denominator in (2.6).

Algorithm 4.1 The SIS algorithm for recursive estimation of the joint posterior.

Sample N samples from the a priori density $P(\boldsymbol{\theta}, \boldsymbol{x}_0)$
for $i = 1$ to N **do**
  $\tilde{w}_0^i = \frac{1}{N}$
end for
for $k = 0$ to T **do**
  for $i = 1$ to N **do**
    Sample $\boldsymbol{x}_k^i$ from $q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}^i, \boldsymbol{\theta}^i)$
    Assign the particle weight: $w_k^i = \frac{P(\boldsymbol{z}_k \mid \boldsymbol{\theta}^i, \boldsymbol{x}_k^i) P(\boldsymbol{x}_k^i \mid \boldsymbol{\theta}^i, \boldsymbol{x}_{k-1}^i)}{q(\boldsymbol{x}_k^i | \boldsymbol{x}_{k-1}^i, \boldsymbol{\theta}^i, \boldsymbol{z}_k)} \tilde{w}_{k-1}^i$
  end for
  Normalize the weights:
  for $i = 1$ to N **do**
    $\tilde{w}_k^i = \frac{w_k^i}{\sum_{i=1}^N w_k^i}$
  end for
end for

4.3 Choosing the proposal density

Section 3.5.3 discussed the performance of Monte Carlo methods and in particular the consequence of the choice of a proposal density on this performance. The same principles still hold for sequential importance sampling: the choice of the importance sampling density strongly influences the posterior variance of the Monte Carlo estimator. (Doucet, Godsill, and Andrieu 2000), proposition 2, proves that the optimal (incremental) importance function for a Markovian system is

$$\pi(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k}) = P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k). \tag{4.7}$$

In that case, eq. (4.5) becomes

$$\begin{aligned} w_k(\boldsymbol{x}_k, \boldsymbol{\theta}) &\sim \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k)} w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) \\ &=^1 \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}) P(\boldsymbol{z}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta})} w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) \\ &= P(\boldsymbol{z}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}) w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}), \end{aligned} \tag{4.8}$$

where 1 follows from applying Bayes' rule to $P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k)$.

So, in this case, the incremental weight is the probability of measuring the

measurement at timestep k, given the state (of the particle) at time $k-1$:

$$\hat{w}_k = P(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) = \int P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) d\boldsymbol{x}_k, \tag{4.9}$$

if $\boldsymbol{x}$ is continuous. Except for particular cases discussed later in this Chapter, this integral is time-consuming to calculate. Furthermore, in the case of the optimal importance density, the incremental proposal density is

$$P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k) \sim P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}). \tag{4.10}$$

Unfortunately, this density is very hard to sample from in most cases. Indeed, as soon as the measurement model contains non-linearities, the proposal function contains a non-linear function $\boldsymbol{h}(\boldsymbol{x}_k)$ which renders exact sampling very hard.

Many Particle Filter algorithms differ only by the proposal density they use. Some of them make use of specific assumptions about the nature of $\boldsymbol{x}$ and/or $\boldsymbol{\theta}$, these are discussed in the following subsections.

4.3.1 The optimal proposal density

As can be seen from eq. (4.10), the optimal proposal density in the case of filtering is proportional to the product of the measurement model and the system model (expressed as a function of the unknown parameters $\boldsymbol{x}_k$ and evaluated at the current particle state $\boldsymbol{x}_{k-1}, \boldsymbol{\theta}$ and the current measurement $\boldsymbol{z}_k$. This means that, in the case where model uncertainty is assumed to be additive Gaussian—which is the case for most applications dealt with in this thesis and in robotics in general—, the optimal proposal can be calculated if the *measurement model* is linear. In that case, if

$$\begin{aligned} P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) &= \mathcal{N}\left(\boldsymbol{f}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}\right), \boldsymbol{\Sigma}_s\right), \\ P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}_k) &= \mathcal{N}\left(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{\Sigma}_m\right), \end{aligned} \tag{4.11}$$

where $\boldsymbol{f}$ represents the non-linear system equation and $\boldsymbol{H}$ expresses the linear measurement equation, the proposal density becomes

$$P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k) = \mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right), \tag{4.12}$$

where

$$\begin{aligned} \boldsymbol{\Sigma}^{-1} &= \boldsymbol{\Sigma}_s^{-1} + \boldsymbol{H}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{H}, \\ \boldsymbol{\mu} &= \boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_s^{-1} \boldsymbol{f}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}\right) + \boldsymbol{H}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{z}_k\right), \end{aligned} \tag{4.13}$$

and the incremental weight update factor is

$$\hat{w}_k = P(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}) \sim \mathcal{N}\left(\boldsymbol{H}\boldsymbol{f}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}\right), \boldsymbol{\Sigma}_m + \boldsymbol{H}^T \boldsymbol{\Sigma}_s \boldsymbol{H}\right). \tag{4.14}$$

This derivation is presented for fully continuous state vectors in (Doucet 1998).

Applications of this proposal density are typically found in autoregressive $AR(k)$ (Geweke 1989; Fearnhead 1998) or time-varying autoregressive $TVAR(k)$ models (Godsill and Clapp 2001; Andrieu, Davy, and Doucet 2003; Andrieu, Doucet, Singh, and Tadic 2004), and in Jump Markov (Linear) Models (Akashi and Kumamoto 1977; Doucet, Gordon, and Krishnamurthy 2001; Andrieu, Davy, and Doucet 2003; De Freitas, Dearden, Hutter, Morales-Menéndez, Mutch, and Poole 2004) (these models are described in Chapter 5).

4.3.2 Using an analytic filter in the proposal step

Many real-life problems cannot be successfully modeled by a linear measurement equation. To incorporate the information from the measurement, another filter (consuming far less resources than a PF) can be used in the proposal step. Indeed, e.g. for systems with additive Gaussian uncertainty but with a non-linear measurement equation, e.g. a linear approximation of the measurement equation can be made around the prediction of the system model $\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta})$. This method is sometimes referred to as the *Extended Kalman Particle Filter* (EKPF) (Doucet 1998).

Note that linearisation of the measurement equation in order to obtain a *proposal* density does not imply the disadvantages of KF linearisation techniques!

Other variations found in literature are e.g. the *Sigma Point Particle Filter* and the *Gaussian Mixture Sigma Point Particle Filter* (van der Merwe and Wan 2003). These are Particle Filters that use a *Linear Regression Kalman Filter* (Lefebvre 2003) in their proposal step. Worth mentioning in this scope is the *Unscented Particle Filter* (van der Merwe, Doucet, de Freytas, and Wan 2000; Julier and Uhlmann 2004): The Unscented Kalman Filter (UKF) (Julier and Uhlmann 1997) is known to always generate *consistent* estimates, at the cost of a sometimes rather uninformative state estimate (Lefebvre, Bruyninckx, and De Schutter 2004b). This makes it extremely valuable since it generates proposal densities with heavier tails than the true posterior (See Section 3.5.4). This reduces the Monte Carlo variance of the resulting estimator, or allows to obtain the same variance with less samples (i.e. faster).

As will be demonstrated in Chapter 6, the large computational complexity is still a problem for the ACM tasks dealt with in this thesis, especially if larger uncertainties are considered. Therefore, the use of the above described filters for estimation in ACM is a topic of future research.

4.3.3 Other proposal densities

A variety of other proposal densities are found in literature. Some of them are generally applicable, such as the *Auxiliary Sampling Importance Resampling*

(ASIR) filter (Pitt and Shephard 1999), others (described below) are problem specific. The key idea of the ASIR filter is based on the observation that, in the case of the optimal importance density, the weights at timestep k are independent of the particle particle values at timestep k (See eq. (4.9)). This means that the *selection step* (described in the next Section) can be performed *before* extending $\boldsymbol{x}_{1:k-1}$ to $\boldsymbol{x}_{1:k}$, thereby reducing the Monte Carlo variation of the weights.[4] However, as the optimal proposal density typically cannot be calculated for most non-linear (measurement) models, the ASIR filter typically uses an approximation of the optimal proposal density:

$$P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k) \sim P(\boldsymbol{z}_k|\boldsymbol{\mu}_k, \boldsymbol{\theta})P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}), \tag{4.15}$$

thereby avoiding the nonlinear dependency on $\boldsymbol{x}_k$ of the proposal density. $\boldsymbol{\mu}_k$ can be determined depending on the problem, but a typical example is $\boldsymbol{\mu}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta})$. The ASIR filter actually updates the weight of the particles at timestep $k-1$ by multiplying them with $P(\boldsymbol{z}_k|\boldsymbol{\mu}_k, \boldsymbol{\theta})$. Then, a discrete variable (the auxiliary variable) is sampled from the discrete set of particles with the updated weights, and that particle is used in the proposal density $P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})$, followed by a normal SIS step with the system model as proposal density. This approach implies that particles that would probably have a small weight after updating, are discarded *before* the actual proposal step. The ASIR filter outperforms Particle Filters using only the system model as proposal in the case of severe outlier measurements *and* if the approximation $P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta}) \approx P(\boldsymbol{z}_k|\boldsymbol{\mu}_k, \boldsymbol{\theta})$ is valid.

This thesis (and other research, e.g. (Fox, Burgard, and Thrun 1999)) uses the system model as a proposal density

$$q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k) = P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}), \tag{4.16}$$

which does not take the measurements into account in the proposal density. Indeed, the main disadvantages of the methods above described are the fact that (i) they result in a more complex (and thus error prone) modeling process and (ii) the resulting filters are more time- and resource consuming than a standard Particle Filter for a given number of particles. E.g. the filters described in the previous section run an analytic filter for each of the particles: An Extended Kalman Particle Filter with 1000 particles performs 1000 update steps of an Extended Kalman Filter at each timestep! Also, for easy problems with rather "well behaved"[5] measurements, a standard Particle Filter using

[4] An ad hoc application of this principle, in which a sample is propagated through the system model, and rejected if its weight does not exceed a threshold, was already proposed for the Bootstrap Filter algorithm in (Gordon, Salmond, and Smith 1993) and coined as *prior editing*.

[5] i.e. when the correction step of a KF algorithm does not alter the prediction estimate significantly.

the system density as proposal typically outperforms a more complex filter (taking into account measurement information). Indeed, for a given interval, it can use far more particles than the complex filter and the benefits of taking into account the extra information are negligible in this case.

However, when increasing initial uncertainties, filters that do not take into account measurement information will eventually diverge. One possibility to overcome this is to run a more complex filter as long as the estimates have not "converged" (e.g. during what is called global localization in mobile robotics), and use a simpler filter once the estimate is accurate enough (e.g. for tracking purposes). So, the choice of better proposal densities is certainly a topic for further research in autonomous compliant motion problems.

However, in general, there seems to be no solution that is applicable to each and every problem and problem specific trial and error (based on the initial uncertainty, model uncertainty, number of particles, ...) is necessary to determine the best option for a given case.

An example of a problem specific proposal density is (Lenser and Veloso 2000), used for global localisation in mobile robotics. As long as the robot is certain about its position estimate—the (unnormalized) weight of the samples after updating the filter exceeds a user defined threshold—, the system model is used as a proposal density. If the total (unnormalized) weight of the samples after updating drops below the threshold, a certain percentage of the samples is generated from the measurement model $P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})$. As the measurement model is non-linear, samples from the measurement model are generated using the rejection sampling technique (non-realtime).

4.4 Impoverishment

The algorithm as described in algorithm 4.1 is mostly referred to as SIS or sequential imputation (Liu and Chen 1995). Unfortunately, as k increases, the weights become very unevenly distributed. This phenomenon is called *impoverishment* (also referred to as *degeneracy* or *depletion*) and caused by the fact that, as time increases, the target and the proposal density grow further apart. After a certain number of measurements, one particle carries all the weight, and all the others have a zero weight.

Moreover, since Particle Filters use a *discrete approximation* of the posterior density, they are also more sensitive to outliers. When there's an outlier present, particle weights are very unevenly distributed, and it takes a lot of samples to obtain a "good" estimate of the true posterior density. A similar problem occurs with very accurate measurement models. This is illustrated in figure 4.1. A good choice of (incremental) proposal density can slow down this problem, but cannot avoid it. The following subsections describe some possibilities to deal with impoverishment.

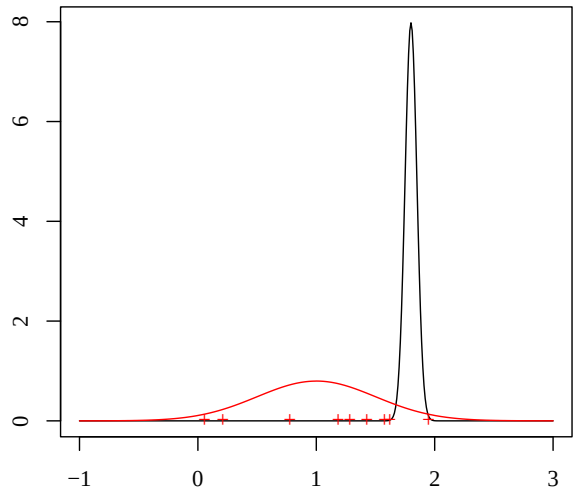


FIGURE 4.1: Illustration of the peaked likelihood degeneracy problem of Particle Filters in 1D. When the likelihood is too peaked, all the samples will get low weights. The problem gets worse for high–dimensional state spaces and when the initial uncertainty is large.

4.4.1 Resampling

Resampling[6] from the discrete sample set is one possibility of dealing with impoverishment. It can be achieved in $\mathcal{O}(N)$ by sampling from the discrete set $\{(\boldsymbol{x}^i, \boldsymbol{\theta}^i), \hat{w}^i\}$ (Gordon, Salmond, and Smith 1993) (Section 3.3). Contrary to algorithm 3.4 for generating samples using importance sampling, the number of samples is usually kept constant in this case, due to computational constraints.

However, resampling also comes at a cost: the above procedure increases the Monte Carlo variance of the resulting estimators. Therefore, it's important to perform the resampling step as infrequently as possible. Earlier research often used *fixed period resampling* (with resampling every time step as a special case). This has two disadvantages:

- The Monte Carlo variance is sometimes unnecessarily increased by resampling;
- Determining the resampling period is only possible by trial and error.

[6] An enormous variety of names is used to denote the resampling process, e.g. *reweighting*, *pruning*, *reconfiguration*, *(re)juvenation*—i.e. resampling plus a MCMC move step—, *selection*, *branching*, ...

Therefore, nowadays mostly *dynamic resampling* is applied. This procedure uses a criterion, typically the Effective Sample Size (ESS) described in Section 3.5.3, to determine if it's necessary to perform a resampling step at timestep k. Note that the calculation of the properties of the posterior PDF, always should be done *before* resampling is performed, as this results in a lower variance estimator.

To avoid the extra variance that is introduced by the multinomial resampling process, other alternatives (often denoted as *selection schemes*) have been developed:

- *Residual resampling* (Liu and Chen 1998), is a near deterministic sampling algorithm, in which each new sample in the set is represented by $\lfloor N\hat{w}_k^i \rfloor$ samples. N denotes the original number of samples, and the $\lfloor\rfloor$ operator takes the integer part of its argument. The remaining $N - \sum_{i=1}^{N} \lfloor N\hat{w}_k^i \rfloor$ are generated by standard multinomial resampling. This algorithm results in a lower Monte Carlo variance and is also less computationally expensive than full multinomial resampling.

- *Stratified resampling* (Kitagawa 1996). Instead of generating N (ordered) random samples on the interval $[0, 1]$, this algorithm divides the interval $[0, 1]$ into N subintervals (the "strata") and generates exactly one sample in each of the intervals. It can be easily shown that this reduces the resulting Monte Carlo variance of the resampling stage (see e.g. (Fearnhead 1998), Section 5.1).

- *Deterministic resampling* (Kitagawa 1996). This algorithm does not use *random* samples on the $[0, 1]$ interval for resampling but chooses

$$u^j = \frac{j - \alpha}{N}, \quad \text{for fixed } \alpha \in [0, 1).$$

  Although the resulting algorithm is not random anymore, good results have been obtained with this (fast) algorithm.

- *Systematic or Minimum variance resampling* (Carpenter, Clifford, and Fearnhead 1999a; Fearnhead 1998) is a generalization of the residual sampling algorithm, in which the randomness is obtained by shuffling the samples before applying the deterministic step. Since there is no multinomial sampling performed anymore for any sample, the resulting Monte Carlo variance is less than that of residual sampling.

(Carpenter, Clifford, and Fearnhead 1999a) propose to use one of the above techniques in combination with modified weights. E.g. the original weights are first replaced by their square root and then one of the above resampling procedures is applied. This is referred to as *resampling with tempered weights*.

The Bootstrap Filter The SIS algorithm that uses the system equation as its proposal density and incorporates a resampling stage is the most used Sequential Monte Carlo method and most often referred to as the *Bootstrap Filter*. The algorithm is described as algorithm 4.2.

Algorithm 4.2 The Bootstrap Filter for recursive estimation of the joint posterior: A SIS algorithm that uses the system model as proposal density and includes a resampling step.

Sample N samples from the a priori density $P(\boldsymbol{\theta}, \boldsymbol{x}_0)$
for $i = 1$ to N **do**
  $\tilde{w}_0^i = \frac{1}{N}$
end for
for $k = 0$ to T **do**
  for $i = 1$ to N **do**
    Sample $\boldsymbol{x}_k^i$ from $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}^i, \boldsymbol{\theta}^i)$
    Assign the particle weight: $w_k^i = P(\boldsymbol{z}_k \mid \boldsymbol{\theta}^i, \boldsymbol{x}_k^i)\tilde{w}_{k-1}^i$
  end for
  Normalize the weights:
  for $i = 1$ to N **do**
    $\tilde{w}_k^i = \frac{w_k^i}{\sum_{i=1}^N w_k^i}$
  end for
  Calculate the effective sample size $ESS = \frac{1}{\sum_{i=1}^N (\tilde{w}_k^i)^2}$
  if $ESS < threshold$ **then**
    Resample
  end if
end for

4.4.2 Application of a MCMC step

Resampling is only one way to deal with impoverishment. Unfortunately, if some of the weights are too large after a measurement update, only a few samples will survive the resampling step, and the resulting Monte Carlo variance of the estimator increases disproportionally. In particular, this is the case if the model contains unknown parameters (physical or hyper-parameters, see Chapter 5). In recent years, authors have come up with other techniques to deal with impoverishment, complementary to resampling. One suggestion is to add a MCMC step (Section 3.7) to the Particle Filter algorithm[7] (MacEachern, Clyde, and Liu 1999; Carpenter, Clifford, and Fearnhead 1999b; Gilks and

[7] Typically a MCMC step will be combined with a resampling step.

Berzuini 2001; Berzuini and Gilks 2001). This algorithm is also referred to as the *MHIR* Filter (Metropolis–Hasting Importance Resampler) or *Resample-Move*. Suppose we use the Metropolis–Hastings algorithm (Section 3.7.1) and choose a proposal function

$$q(\boldsymbol{x}_k, \boldsymbol{\theta}, \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}^{(i)}),$$

where the superscript (i) denotes the ith sample, and it is valid for both states and parameters. In this case, the target density is the joint posterior, hence the acceptance probability of a new sample $\boldsymbol{x}_k, \boldsymbol{\theta}$ becomes

$$a = \frac{P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})}{P(\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}^{(i)}|\boldsymbol{z}_{1:k})} \frac{q(\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}^{(i)}, \boldsymbol{x}, \boldsymbol{\theta})}{q(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}^{(i)})}, \tag{4.17}$$

and in the case of a symmetric proposal density

$$a = \frac{P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})}{P(\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}^{(i)}|\boldsymbol{z}_{1:k})}. \tag{4.18}$$

This quantity can be evaluated by applying Bayes' rule for both the numerator and denominator. Unfortunately however, this means that the history of each particle should be stored, which results in linearly increasing memory and time requirements, even in the case of filtering.

Other issues with adding a MCMC step are the choice of a proposal distribution and when to perform the MCMC steps. Chopin (2002) proposes to use a Gaussian kernel with the current sample as mean value and a covariance matrix based on the sample set covariance. This results in a black box kernel which is independent of the used model and ensures rapid mixing asymptotically. Indeed, as time increases,[8] the posterior converges to a Gaussian distribution with mean and covariance matrix estimated by the Particle Filter. The acceptance rate then becomes one and every move is accepted.

The computational requirements of the MCMC steps are $\mathcal{O}(k)$, where k denotes the timestep. However, as time increases and the posterior converges to a unimodal Gaussian, sample depletion decreases. So, MCMC steps can be applied less frequently. Furthermore, once the posterior has converged and does not change significantly anymore, the acceptance ratio will be near 1. This means that, for off-line estimation purposes, adding a MCMC step is feasible, even for large measurement sets. Berzuini and Gilks (2001) propose to apply a MCMC step with probability $\mathcal{O}\left(k^{-\delta}\right)$, with $\delta > 0$.

An alternative to avoid having to store the whole history is to use Gibbs sampling. Unfortunately, as mentioned in Section 3.7.2, finding the full conditional distributions is hard for high dimensional problems, but it can be useful

[8] And provided the Particle Filter converges.

in the case of lower dimensional problems where a lot of hyper-parameters have to be estimated (Fearnhead 1998; MacEachern, Clyde, and Liu 1999; Carpenter, Clifford, and Fearnhead 1999a).

4.4.3 Methods based on rejection control

(Liu and Chen 1998; Liu, Chen, and Wong 1998) propose the combination of sequential importance sampling with rejection sampling methods, which they coin *rejection controlled sequential importance sampling* (RC-SIS). The algorithm is similar to the ASIR filter in the sense that it uses a look-ahead principle: Samples with small weights are rejected an a new sequence of samples is generated. The algorithm is as follows. At time-step k, accept $\boldsymbol{x}_k, \boldsymbol{\theta}$ drawn from the proposal distribution with probability

$$a_k = min\left[1, \frac{w_k(\boldsymbol{x}_{1:k}, \boldsymbol{\theta})}{c}\right], \tag{4.19}$$

where c is a user defined time-varying threshold value. If the sample is accepted, its weight is updated. If the sample is rejected, a new sequence of samples starting from $k = 0$ is generated according to the same method, until a sequence passes all control *checkpoints*. Actually, this boils down to adapting the proposal distribution in the light of current weights.

Of course, this approach is also hard to apply in a fixed-period context and only works well if the proposal distribution approximates the posterior "fairly well" and once the posterior at timestep k, $P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$, and timestep $k+1$, $P(\boldsymbol{x}_{k+1}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k+1})$, are not too far apart. Liu and Chen suggest using a relatively high threshold in the beginning of the estimation when the particle sequences are cheap to generate, and then gradually decrease it once the posterior has converged.

4.5 Convergence of Particle Filters

One can prove that under minimal assumptions (i.c. if the likelihood is bounded and using random resampling schemes), Particle Filters do asymptotically converge to the so-called *optimal filter* (LeGland and Oudjane 2004). Crisan and Doucet (2002) show that the mean square error of the estimator converges to zero with a convergence rate of $\mathcal{O}(1/N)$

$$E[(\hat{I}_k - I_k)^2] \leq \frac{c_k}{N}, \tag{4.20}$$

where I_k is a property of the true posterior at timestep k (the target density from eq. (3.5)), e.g. in the case of filtering:

$$I_k = \int P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{\theta})\, d\boldsymbol{x}_k d\boldsymbol{\theta}. \tag{4.21}$$

$\hat{I}_k$ denotes the Particle Filter's approximation for this property, and c_k is a time-varying constant. However, the above convergence result only holds for bounded functions $\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{\theta})$. This means that, to the best of my knowledge, there are currently no general convergence results for e.g. $E_{P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})}\left[[\boldsymbol{x}_k\boldsymbol{\theta}]^T\right]$!

Another problem with the above theorem is that c_k is a *time-varying* constant. This means that, in practice, to ensure a given precision, the number of samples has to be increased as k increases. (LeGland and Oudjane 2004) demonstrate that, *if* the proposal density only weakly depends on $\boldsymbol{x}_{k-1}, \boldsymbol{\theta}$, c_k remains constant in time. However, the applications in this thesis mainly deal with parameter estimation or combined parameter/state estimation. In that case, the proposal density is always strongly dependent on $\boldsymbol{x}_{k-1}, \boldsymbol{\theta}$. In practice, this can lead to drift in the Particle Filters estimates as observed by (Andrieu, de Freitas, and Doucet 1999).

The above results open a rather pessimistic view to the application of sequential Monte Carlo methods for the estimation of the joint posterior $P(\boldsymbol{x}_k, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$. Nevertheless, this thesis (Chapter 6) demonstrates that, even with basic proposal densities and resampling techniques, acceptable results are obtained with a Particle Filter for the estimation of the geometrical parameters during a sequence of more than 1000 measurements. Moreover, as Particle Filtering still is a relatively new field, stronger convergence results can be expected in the future.

4.6 Choosing the number of particles

Apart from the convergence properties for Particle Filters, which hold for $N \to \infty$, practitioners are also (or even more) interested in the issue of "How many samples are necessary to obtain reasonable estimates". Most authors use trial and error to determine the adequate number of particles, just as with "non-sequential" importance sampling (Chopin 2002). However, some research has been done on using a variable number of samples, which can be justified in problems of global localisation and even more in "kidnapped-robot-like" problems. In the latter case, unmodeled events strongly modify the posterior's shape and the posterior cannot cause the measurements anymore.[9]. Fox, Burgard, Dellaert, and Thrun (1999) propose to use a *likelihood-based adaptation* to deal with these situations. During inference, samples are generated until the sum of the weights exceeds a threshold. As longs as the Particle Filter has not converged, many sample weights will be small and a lot of particles are required to describe the posterior accurately. Once con-

[9] One example is to lift up a mobile robot tracking its position accurately and put it down somewhere else in the map, from where the name kidnapped-robot originates.

verged,[10] most weights are large and few samples are necessary to reach the threshold.

However, this approach fails to make a distinction between a unimodal and a multi-modal posterior. (Fox 2001; Fox 2003) use an algorithm they call *KLD-Sampling*. The algorithm calculates the required number of samples at each iteration, based on an approximation of the Kullback-Leibler distance between the true and estimated posterior. It uses the predictive belief state $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k-1})$ as an approximation of the true posterior and the χ^2 approximation is in principle only valid for $N \to \infty$. However, the approach has experimentally been verified and was found to perform better than standard SIR and the above described likelihood-adaptive Particle Filters in a mobile robot localisation problem.

4.7 Software and hardware implementations

There is quite some software available implementing Sequential Monte Carlo methods. Unfortunately, most of it is rather ad hoc, undocumented and hard to reuse. This Section provides pointers to some reusable and well documented implementations, without being exhaustive:

- ReBEL (Recursive Bayesian Estimation Library) (van der Merwe) is a Matlab toolbox for sequential Bayesian estimation in state space models. It is free only for academic and/or non-commercial use.

- BFL (Bayesian Filtering library) (Gadeyne 2001b) is an open source C^{++} library with support for several Particle Filter algorithms with different proposal densities and resampling schemes. Any PF algorithm fits in the BFL framework. Bayes^{++} (Stevens 2003) is also an open source C^{++} Bayesian estimation library with support for the SIR algorithm. Both are discussed in Chapter 7.

- Recently, research has been conducted about parallel hardware implementation of Sequential Monte Carlo methods on Field-Programmable Gate Arrays (FPGAs) (Bolic 2004; Bolic, Djuric, and Hong 2004). The bottleneck for parallel implementation is the distribution of the resampling stage.[11]

[10] Mostly this means that the posterior is a unimodal density with small covariance matrix and entropy.

[11] Indeed, as can be seen from algorithm 3.1 and 3.2, the Cumulative Density Function is necessary for resampling, which means that all (distributed) weights must be centralized for resampling.

4.8 Conclusions

This Chapter provides a literature survey of the relatively new field of Sequential Monte Carlo methods research. It tries to give a bird's eye view of the field of Particle Filtering since (Gordon, Salmond, and Smith 1993). Particle Filtering is a very promising technique to deal with any kind of model, and computing resources are continuously increasing. However, this Chapter demonstrates that the Particle Filter algorithm is far from being fool proof, and requires skilled application. The choice of the best proposal density for a given model is very often problem dependent, as is the resampling scheme and the criterion when to resample. The choice of the appropriate number of particles is often only possible by trial and error and there are no general convergence results yet.

This thesis uses Particle Filters to estimate the joint posterior $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$. The next Chapter compares Particle Filter techniques with other Filtering algorithms for particular cases of the joint posterior, and argues why Particle Filters are appropriate to deal with the hybrid autonomous compliant motion estimation problems dealt with in this thesis.

Chapter 5

Classification of Bayesian inference algorithms

5.1 Introduction

This Chapter provides a classification of Bayesian algorithms for recursive inference on the joint posterior density $P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k})$, *based on the nature of the parameter and state vector* (discrete, continuous or hybrid). The focus of the Chapter is on

- applications in robotics in general and autonomous compliant motion in particular;
- state-of-the-art Monte Carlo techniques.

The Chapter provides a classification, that helps to choose what algorithms are *applicable and most appropriate* once a particular model is chosen. For a given subclass, the overview also clarifies the relationship between the choice of models and the computational complexity of (online) algorithm implementations. The models are illustrated with (dynamic) Bayesian networks (Section 2.6). Furthermore, this classification also shows that estimation research currently only focusses on a *subset* of available models and describes some new graphical models previously unused in estimation. These models are used in the ACM estimation problem described in the next Chapter. This Chapter also describes the link between these models and the models used in literature for solving discrete *data association* problems and shows how implicit measurement equations can be considered as *continuous data association* problems.

Last but not least the classification also provides the foundations for the BFL library discussed in Chapter 7. This overview is complementary in scope

with those of Murphy (2002b) and Minka (2004), who base their classification on the *representation of the posterior*. The Chapter focusses on *recursive filtering*, although batch methods are also considered. Where appropriate, pointers to non-Bayesian algorithms for dealing with uncertainty are given and the links with Bayesian algorithms are explained.

Table 5.1 provides a bird's eye view on the classification. Section 5.2 discusses the general interpretation of the classification, while the rest of the sections discuss one or more aspects of the classification in more detail.

5.2 Classification of the joint posterior based on the nature of $\boldsymbol{x}, \boldsymbol{\theta}$

Murphy (2002b) and Minka (2004) present an overview of (recursive) Bayesian estimation algorithms, *based on the representation of the posterior density*. Roughly,[1] the following cases can be identified in case of online Bayesian inference.

1. **Analytical posterior** The posterior density is represented by an analytical function. The best known examples are the approximation of continuous posterior as a Gaussian, or a sum of Gaussians, but all algorithms based on the principle of conjugacy (Section 2.5.2) belong to this category. By definition, analytical approaches cannot handle discrete Random Variables. Several Filters in parallel are used to estimate hybrid Random Variables.

2. **Grid based posterior** The posterior is represented by a uniform grid. Grid based approaches can handle all types of RVs. However, this approach scales very badly as the dimension of the states/parameters increases. If the granularity of this grid is time-varying, the term *adaptive grid* is sometimes used.

3. **Sample based posterior** (Chapters 3 and 4). The posterior is represented by a set of *random* samples. Therefore this approach is sometimes referred to as the *randomized adaptive grid* (Doucet, Gordon, and Krishnamurthy 2001) approximation of the posterior. Just as grid based approaches, they can handle any type of RVs but only represent parts of the state space that are relevant. Therefore this approach scales better as the dimension of the state space increases. This Chapter focusses on Monte Carlo methods, as they are best suited to deal with the multi-dimensional and hybrid applications in this thesis.

[1] Some techniques *combine* some of this representation, e.g. Rao-Blackwellization represents some dimensions of the posterior PDF as samples, and other dimensions analytically.

Fully Bayesian algorithms use prior information and provide an (approximative) description of the posterior *PDF*. To reduce the computational complexity of the estimation process, approximative algorithms, such as the *Maximum Likelihood* (ML) and *Maximum A Posteriori* (MAP) algorithms (Section 2.5.1), that are not fully Bayesian, have also been developed, reducing PDFs into *point estimates*. Typical examples are the Viterbi algorithm for inference in Hidden Markov Models (HMMs) (Forney 1973; Rabiner 1989), which yields the MAP value of the posterior, and the Expectation-Maximization algorithm (EM) (Dempster, Laird, and Rubin 1977) (Section 5.5.1), which uses a *hill-climbing* method to provide a point estimate of (a part of) the parameter vector off-line, before estimating the rest of the unknown variables online with a fully Bayesian method.

As mentioned, not all algorithms can deal with a state or parameter vector that is either discrete or hybrid, e.g. the standard (Iterated Extended) Kalman Filter cannot handle hybrid RVs, and a Bootstrap Filter has trouble with the estimation of pure parameters. Therefore, table 5.1 provides a 2D classification of Bayesian techniques, models and applications from another point of view: A classification of the joint posterior *based on the nature of* $\boldsymbol{x}$ *and* $\boldsymbol{\theta}$. Together with the classification based on the representation of the posterior, this results in a 3D table.

The table contains 15 ($4 \times 4 - 1$—in case both $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are known—) distinct cases of possible $\boldsymbol{x}/\boldsymbol{\theta}$ combinations. The first column contains all cases where there are no unknown state random variables in the model: these are pure (recursive) parameter estimation problems (Section 5.4), which can be modeled by (static) Bayesian networks (BN). The first row contains all cases where $\boldsymbol{\theta}$ is (supposed to be) known. These are pure state estimation problems (Section 5.3). All problems that involve the recursive estimation of unknown parameters/states can be modeled by Dynamic Bayesian Networks (DBN).

Model selection, a term frequently found in literature, applies to all cases in which $\boldsymbol{x}$ or $\boldsymbol{\theta}$ contains a discrete part (is either discrete or hybrid). If the state vector $\boldsymbol{x}$ is (partly) discrete, this is often referred to as *switching model selection*. A typical example is the discrete state estimation in a HMM. If the parameter vector $\boldsymbol{\theta}$ is discrete, the model selection process is known as *hypothesis testing*. For example, a robot that has to be able to distinguish work pieces in order to perform the appropriate machining tasks is a typical case of hypothesis testing. Obviously, combinations between hypothesis testing and switching model selection are possible. One application is to simultaneously distinguish between different speakers and recognize what they say. In that case, the discrete parameter vector contains entries for the different speakers and the discrete states are the phonemes to be recognized.

$\Theta \setminus X$	Known	Continuous	Discrete	Hybrid	Remarks
Known		Simple State estimation, (non-)linear State Space models	HMM (with known parameters)	Interacting Multiple Model (IMM), Switching State Space Model (SSM), Jump Markov (Non)-(Linear) Model, Markovian Switching systems	
		KF / PF / grid based F/ Gaussian Mixture Models (GMM)/ Information Filter/ Square Root Filter...	Viterbi algorithm (MAP) / PF	IMM filtering, Assumed Density Filtering (ADF), Generalised Pseudo Bayes (GPB, GPB2), Reversible Jump MCMC, Moment Matching, Rao-Blackwellization, JPDAF variants	
		Mob robot localisation with map + known data association, time series (known params)	Speech recognition, fault detection, contact state estimation, feature detection, (dynamic intention estimation)	Tracking of a maneuvering target, Fault diagnosis, Piecewise linearisation, *Mobile robot localisation with map + unknown data association, Multi-target tracking, SLAP*	
Cont.	Simple Parameter estimation	System identification, Data augmentation, incomplete data problems	HMM (with unknown parameters), Classification (supervised learning)		Parameters can be physical or hierarchical (hierarchical Bayes---hyperparameters)
	KF / PF (with care)/ ML methods	KF (Joint KF)/ PF / continuous Expectation Maximization (EM)/ FastSLAM (known data association)	PF /Viterbi + Baum-Welch (discrete EM) or KF	KF / PF / EM/ FastSLAM + data-association (e.g. Nearest Neighbour Filter, ML)/ Rao-Blackwellization	
	Geometrical Parameter estimation given CF	SLAM with known data association	Simultaneous CF recognition + geometrical parameter estimation	*SLAM with unknown data association*	
Discrete	Simple Hypothesis Testing	Multiple model (non interacting) filtering			Hypothesis testing
	Bayes' rule (Bayes Factor, posterior odds)	Hypothesis test (Bayes Factor, posterior odds) + filter	Particle Filters		
	Medical diagnosis, intention estimation	Comparing a discrete number of models for state estimation	Distinguish between 2 speakers + recognise what they say	Multi-target tracking, SLAP with varying number of targets	
Hybrid	Multiple model (non interacting) filtering				Learning, training
	Hypothesis test (Bayes' Factor, posterior odds) / filters				
	Model building from scratch		Distinguish between 2 unmodeled speakers		
	BN	DBN			

Table 5.1: Classification of the joint posterior based on the nature of $\boldsymbol{x}$ and $\boldsymbol{\theta}$. Each cell of the table contains three items: model names, algorithms, and applications.

Learning is a term often found in AI literature. The term is used to denote all estimation problems in which parameters have to be estimated. These can be *hyperparameters*,[2] such as the standard deviation on a force– or distance sensor measurement, or physical parameters, such as the transition probabilities in a HMM. The term *training* is often used if this learning is not done in a recursive way, but rather off-line and mostly not fully Bayesian, as with the Expectation-Maximization algorithm (Section 5.5.1).

5.3 Pure State Estimation, known parameters

In the case of pure state estimation, all parameters (both physical and hyperparameters) are assumed to be known, and the Dynamic Bayesian Network (DBN) in figure 2.3 reduces to figure 5.1.

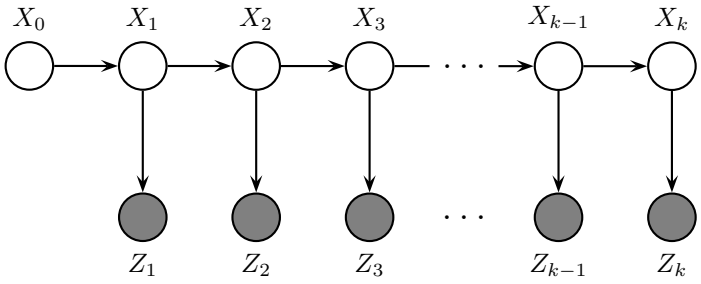


FIGURE 5.1: DBN representing pure state estimation models. The state vector $\boldsymbol{X}_k$ can be continuous, discrete or hybrid.

5.3.1 Continuous states, known parameters

This is probably the most frequent case in literature. An enormous number of variations on the basic Kalman Filter algorithm, that can only deal with linear systems with additive Gaussian uncertainty and Gaussian priors, has been developed.

The best known extensions are probably the *Extended Kalman Filter* (EKF), using a first order Taylor series approximation to linearize the models around the current estimate, and the *Iterated Extended Kalman Filter*

[2] A hyperparameter represents the Bayesian way of dealing with unknown model parameters. When the value of a certain parameter of the stochastical model is also unknown, Bayesian probability theory considers the parameter as *yet another unknown variable*. More info about hyperparameters is found in e.g. (Jaynes 1996).

(IEKF), an iterative form of the latter. *The Information Filter* (Farook and Bruder 1990) uses the *information* matrix—the inverse of the covariance matrix—in its update algorithm, which for some problems[3] results in a more efficient algorithm (sparse Matrices). The *Thin Junction Tree Filter* (TJTF) and the *Sparse Extended Information Filter* (SEIF) belong to this category. The *Square Root Kalman Filter* (Kaminsky, Bryson, and Schmidt 1971; Bierman 1974; Anderson and Moore 1979) uses the square root of the covariance matrix in order to obtain numerically more stable results. *Gaussian Sum Filters* (Alspach and Sorenson 1972) approximate the posterior as a mixture of Gaussians.

Assumed density Filters (ADFs) (a.k.a. moment matching or *weak marginalization*) (Maybeck 1982; Boyen and Koller 1998; Minka 2001) propagate the posterior *exactly* for one time-step, and then approximate it by a well chosen PDF (e.g. a Gaussian mixture) using a certain minimum distance criterion such as the KL-distance. The Expectation–Propagation algorithm (Minka 2001) is a batch extension to these filters, using an iteration step in order to reduce the Kullback-Leibler divergence between the final posterior and its approximation. All above filters typically require analytic or numerical solution of integrals and products of functions.

Sigma Point Kalman Filter variants (including the unscented Kalman Filter) (Julier and Uhlmann 1997; Julier and Uhlmann 2004) propagate a number of well chosen *sigma points* through the system and measurement models, and use these values to calculate the desired characteristics of the posterior. They are sometimes referred to as *Linear Regression Kalman Filters* (Lefebvre, Bruyninckx, and De Schutter 2004b).

Daum (1988) proposes an exact filter that is applicable to a subclass of possible models using exponential PDFs to represent the posterior. Unfortunately, it is hard to identify if a particular model can be estimated with Daum's filter.

The *Non-Minimal State Kalman Filter* (NMSKF) (Lefebvre, Gadeyne, Bruyninckx, and De Schutter 2003) transforms the original state into a higher dimensional space, where the measurement equations are linear and therefore avoids the *accumulation* of linearisation errors which is typical of the above described filters.[4] Unfortunately, it is only applicable to a subclass of all systems (mainly parameter estimation). This thesis does not go into detail about the similarities and differences between all the above algorithms. Bierman and Thornton (1977) and Lefebvre, Bruyninckx, and De Schutter (2004b) provide comparisons between the different variants.

[3] SLAM is a typical example.

[4] Another way to look at this is the fact that, contrary to all previous filters (except somewhat the Thin Junction Tree Filter), the estimate delivered by the NMSKF is not influenced by the *ordering* in which the measurements are processed.

Grid based filters or *Histogram Filters* (Bucy and Youssef 1974; Kitagawa 1987; Tanizaki 1993; Bølviken and Storvik 2001) are only applicable to models where the state vector is low dimensional.[5] *Particle Filter variants* are described in detail in Chapter 4.

Applications of this type of model abound: (global) mobile robot localization using a known map (and assuming known data association (Section 5.3.4). (Fox, Burgard, and Thrun 1998), target tracking (Bar-Shalom and Li 1993), time series estimation (Daum 1988), ... In most of the applications dealt with in this thesis however, the kinematic and dynamic behavior of our serial manipulators or machining tools are assumed to be known and modeled accurately enough to suppose the position and orientation known.

5.3.2 Discrete states, known parameters

This is the case commonly referred to as Hidden Markov Models (HMM) (Rabiner and Juang 1986; Rabiner 1989; Dugad and Desai 1996),[6] where all model parameters are known. The basic HMM topology is shown in figure 5.1. It assumes that the probability for moving from one discrete state at timestep k to another one at timestep $k+1$ is a constant

$$P(x_{k+1} = j | x_k = i) = a_{ij}, \tag{5.1}$$

where i, j are two of the possible discrete states.[7] Furthermore, its observations are also discrete and modeled similarly:

$$P(z_{k+1} = k | x_k = i) = b_{ik}. \tag{5.2}$$

A lot of variations and extensions to this basic model have been developed in literature to allow more powerful and accurate models, such as the Variable Duration HMM (Ferguson 1980; Levinson 1986a; Levinson 1986b) which loosens assumption (5.1), n^{th}-order HMMs (Aycard, Mari, and Washington 2004) that assume the state transition model follows a nth-order Markov chain, Hierarchical HMMs (Murphy 2002b) where the discrete state consists of several discrete parts and one part causes the other (i.e. as described in detail in Section 5.3.3, but with fully discrete states), ...

The Viterbi algorithm (Forney 1973) provides a *Maximum A Posteriori* (MAP) estimate of the (discrete) posterior density. Particle Filters, contrary to Kalman Filter variants that cannot deal with *discrete* state or parameter variables, can be used to estimate the *full* posterior density.

[5] Note that the grid based discretisation of the posterior density, the system and measurement model results in a Hidden Markov Model, see 5.3.2.

[6] Note that a few authors use the term Hidden Markov Models to denote all cases described in this Section, i.e. all models described by the first row of table 5.1.

[7] This corresponds to a *first* order Markov Chain process.

Speech recognition (Rabiner 1989) is definitely the most frequent application of this type of fully discrete state space models, although HMMs have been used for a variety of discrete state estimation problems such as feature detection (Aycard, Mari, and Washington 2004), contact state estimation during assembly (Hovland and McCarragher 1998), subtask identification during telemanipulation (Hannaford and Lee 1991), the identification of human grasping sequences for Programming by Demonstration (Bernardin, Ogawara, Ikeuchi, and Dillmann 2005) and fault detection (Smyth 1994).

5.3.3 Hybrid states, known parameters

In this case, the RV state variable $\boldsymbol{X}$ can be split up into a discrete part $\boldsymbol{X}^d$ and a continuous part $\boldsymbol{X}^c$. Although this hybrid structure could also be modeled by figure 5.1, mostly it is useful to split up $\boldsymbol{X}$ into parts, due to the causal relationship between the discrete and continuous parts. A general DBN representing this case is shown in figure 5.2. In this case, the system

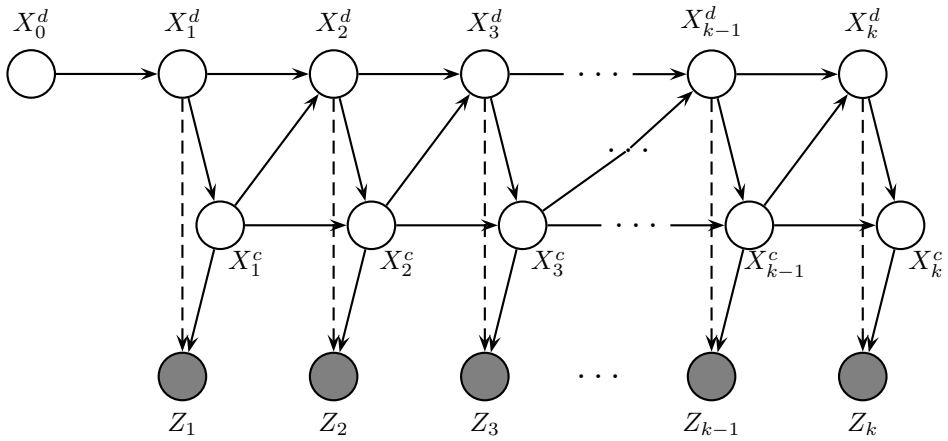


Figure 5.2: Jump Markov System model with known parameters and cross-dependency between the discrete and continuous part. The state vector $\boldsymbol{X}$ is split up into a discrete part $\boldsymbol{X}^d$ and a continuous part $\boldsymbol{X}^c$. The dashed arrow from $\boldsymbol{X}^d$ to $\boldsymbol{Z}$ denotes that this causal relationship is optional.

can be described by the following PDFs:

$$P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d, \boldsymbol{x}_{k-1}^c),$$
$$P(\boldsymbol{x}_k^c|\boldsymbol{x}_{k-1}^c, \boldsymbol{x}_k^d),$$
$$P(\boldsymbol{z}_k^c|\boldsymbol{x}_k^c, [\boldsymbol{x}_k^d]).$$

A typical application of this type of models would be a mobile robot localizing itself in an office building, using a map of different rooms and knowing how to get from one room to another one. The discrete part of the state vector represents the room in which the robot is currently located, the continuous part represents the robot's location in a certain room. $P(\boldsymbol{z}_k^c|\boldsymbol{x}_k^c, \boldsymbol{x}_k^d)$ represents the map of a certain room, $P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d, \boldsymbol{x}_{k-1}^c)$ describes the probability a robot moves to another room given the current room and its position in that room, and $P(\boldsymbol{x}_k^c|\boldsymbol{x}_{k-1}^c, \boldsymbol{x}_k^d)$ describes the motion model in a particular room. This could be useful if e.g. the floor of one room is different from another one, resulting in different motion behavior in different rooms.

Fully analytical approaches will typically fail in such situations, while Particle Filters can handle such situations due to their ability to deal with hybrid RVs.

However, to the best of my knowledge, in the applications and models described in literature, the continuous part does *not* influence the evolution of the discrete part of the state vector. The corresponding models are often denoted as Jump Markov Systems (JMSs). E.g. fault diagnosis is often modeled by a JMS. The discrete state represents the state of the system and is modeled by a first order Markov chain. The system and/or measurement models for the continuous state then depend on the value of the discrete state. However, in a JMS, the value of the continuous state cannot influence the discrete transition behavior. Suppose a JMS is used to describe an axis, where the discrete part of the state vector is one of {`Running,Stopped,InError`} and the continuous state represents the position $\boldsymbol{x}$ along the axis. The probability of going from the {`Running`} state to the {`InError`} state will probably increase if we know that the motor is approaching its end limit with a large speed. This dependency cannot be expressed by a JMS however.

Jump Markov Systems

The model discussed in most literature is the (first order) *Jump Markov (Linear) System* (Harrison and Stevens 1976). This model is shown in figure 5.3 and appears under an enormous number of names in different fields: *Switching State Space* model, *Markovian Switching systems*, *Interacting Multiple* models, ... It is described by the following PDFs:

$$\begin{aligned} P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d), \\ P(\boldsymbol{x}_k^c|\boldsymbol{x}_{k-1}^c, [\boldsymbol{x}_k^d]), \\ P(\boldsymbol{z}_k^c|\boldsymbol{x}_k^c, [\boldsymbol{x}_k^d]), \end{aligned}$$

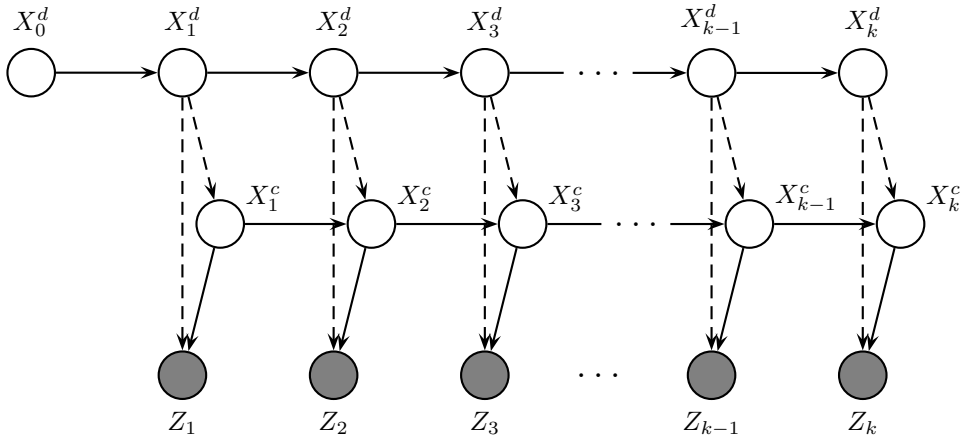


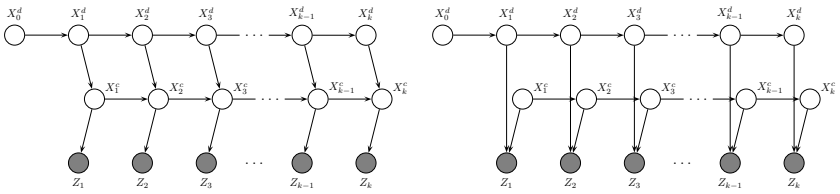
Figure 5.3: Jump Markov System model with known parameters. The state vector $\boldsymbol{X}$ is split up into a discrete part $\boldsymbol{X}^d$ and a continuous part $\boldsymbol{X}^c$. The dashed arrow from $\boldsymbol{X}^d$ to $\boldsymbol{X}^c$ or $\boldsymbol{Z}$ denotes that this causal relationship is optional. This means the discrete state will typically influence the system model, the observation model, or both.

where [ ] denotes an optional argument (reflected by the dashed arrow in figure 5.3).[8] So the evolution of the discrete part of the state vector is governed by a first order Markov Chain. The system and/or measurement model used depends on the state of the discrete part. In case the system and measurement models are linear, these models are called *Jump Markov Linear Systems* or *Switching Kalman Filter models*.

A variety of algorithms exist to solve these models, mostly extensions to the analytical algorithms described in the previous Section. Many algorithms make specific assumptions about the Jump Markov Systems, based on the dashed arrows in figure 5.3 and depending on the application:

- In some cases, the discrete part only influences the evolution of the *continuous part* and the measurement model does not depend on $\boldsymbol{X}^d$. This is shown in figure 5.4(a). Typical applications are a piecewise linearisation of non-linear system models (Ghahramani and Hinton 1998) (a degenerate case of the EKF algorithm), maneuvering target tracking (Bar-Shalom and Li 1993), and realtime (fault) diagnosis (Willsky and Jones 1976; Willsky 1976; Verman, Gordon, Simmons, and

[8] Typically at least one of the dashed arrow series will be present: if none of the dashed arrows is present, the discrete part has no influence on the system or the observation model, and we end up with a non switching model.



(a) Switching Dynamics. (b) Switching Observations.

FIGURE 5.4: Two possible cases of Jump Markov Systems.

Thrun 2004; De Freitas, Dearden, Hutter, Morales-Menéndez, Mutch, and Poole 2004).

- Another case occurs when the discrete part only influences the *measurement* equation (figure 5.4(b)). These models are typically used to obtain a piecewise linearisation of the measurement model (Ghahramani and Hinton 1998), or to deal with outliers or sensor failure (Willsky and Jones 1976; Willsky 1976).

  If the continuous state consists of several, independent *targets*, more specific algorithms are often used, exploiting the independence properties. In that case, the discrete part of the state vector is used to link a measurement with one of the targets. This problem is often called *data association* and is not specific to pure hybrid state estimation problems. Therefore its applications and algorithms are discussed in more detail in a separate Section 5.3.4.

Of course, combinations of both models are possible, e.g. TVAR models with unknown model order for frequency estimation (Andrieu, Davy, and Doucet 2003).

Analytic Filters. The use of analytic representations for Jump Markov Models is not straightforward. Indeed, suppose a continuous unimodal prior is chosen in one of the possible discrete states and all other possible states have zero probability. Then, a derivation similar to eq. (2.5) leads to

$$\begin{aligned} &P(\boldsymbol{x}_k^c, \boldsymbol{x}_k^d | \boldsymbol{z}_{1:k}) \\ &= P(\boldsymbol{z}_k | \boldsymbol{x}_k^c, [\boldsymbol{x}_k^d]) \int \sum_{\boldsymbol{x}_{k-1}^d} P(\boldsymbol{x}_k^c | \boldsymbol{x}_{k-1}^c, [\boldsymbol{x}_k^d]) P(\boldsymbol{x}_k^d | \boldsymbol{x}_{k-1}^d) P(\boldsymbol{x}_{k-1}^c, \boldsymbol{x}_{k-1}^d | \boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1}^c. \end{aligned} \tag{5.3}$$

Due to the term $P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d)$, the unimodal prior will be propagated to all discrete possible states, resulting in a hybrid posterior with N unimodal Gaussians[9] at timestep one if N is the number of possible discrete states, N^2 at timestep two, ...

So to perform filtering with a fixed amount of memory, the number of Gaussians has to be reduced at each timestep. The *Second Order Generalized Pseudo Bayes* (GPB2) algorithm[10] (Ackerson and Fu 1970) propagates all N modes at timestep k to obtain a posterior with N^2 modes, and reduces them *after* updating to N modes by *moment matching*, which is optimal from a KL-point of view. The *Interacting Multiple Model* (IMM) Filter (Blom 1984; Blom and Bar-Shalom 1988; Bar-Shalom and Li 1993) is analogous to the Gaussian Sum Filter, but takes into account cross model jumps. At timestep k, the multimodal posterior (e.g. with N nodes, where N is the number of possible discrete values $\boldsymbol{X}^d$) is reduced to a single Gaussian, and propagated through the sytem and measurement model to obtain the N-modal posterior at timestep $k+1$. The IMM only uses N KF updates (vs. N^2 for the GPB2 algorithm) at each timestep, with no significant extra accuracy, but unlike the GPB2 algorithm, cannot easily be extended to the smoothing case. Other techniques used for dealing with the growth of parameters are applied less frequently, and include iterative and variational methods, See (Murphy 1998) for an overview.

Another frequently used approximation for switching model selection is to use a single analytical filter and select the most probable model using the *innovation*, which is the difference between the expected measured value and the actual measurement. This approach does not use a hybrid posterior, nor a system model $P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d)$ and will therefore typically fail in situations with a lot of model uncertainty. However, it is the most favorable solution in terms of computing requirements. Typical examples are the *NIS* and *SNIS* tests (Mehra and Peschon 1971; Willsky 1976; Bar-Shalom and Li 1993). These tests are sometimes referred to as *model validation* tests.

Particle Filters. Recently, research has been done on the use of Particle Filters for this type of models. Standard Particle Filtering algorithms apply, since they naturally handle hybrid variables, but optimizations exploiting the DBN structure are possible. For Jump Markov *Linear* Systems, *Rao-Blackwellized Particle Filters* (Montemerlo, Thrun, Koller, and Wegbreit 2002; Murphy and Russell 2001) are a good alternative, since the combination of a Particle Filter for the estimation of the discrete part and a standard Kalman Filter allows a significant posterior Monte Carlo variance reduction.

[9] Assuming a linear JMS.

[10] The GPB/GPB1 or Generalized Pseudo Bayes algorithm does not allow mode transitions.

Indeed, the joint posterior can be factorized as

$$P(\boldsymbol{x}^c_{1:k}, \boldsymbol{x}^d_{1:k}|\boldsymbol{z}_{1:k}) = P(\boldsymbol{x}^c_{1:k}|\boldsymbol{x}^d_{1:k}, \boldsymbol{z}_{1:k})P(\boldsymbol{x}^d_{1:k}|\boldsymbol{z}_{1:k}) \tag{5.4}$$

where $P(\boldsymbol{x}^c_{1:k}|\boldsymbol{x}^d_{1:k}, \boldsymbol{z}_{1:k})$ can easily be estimated by a Kalman Filter. This approach is used in (Doucet, Gordon, and Krishnamurthy 2001; Murphy and Russell 2001). For non-linear systems, Andrieu, Davy, and Doucet (2003) propose the use of a modified ASIR filter with an Unscented Kalman Filter step as proposal density. Optionally, an MCMC step (denoted as *reversible jump MCMC* (Green 1995; Andrieu, de Freitas, and Doucet 1999) can be applied to further reduce degeneracy.

McGinnity and Irwin (2001) provide a performance comparison between an IMM and a Particle Filter on a maneuvering target tracking problem.

5.3.4 Data association

When several similar unknown *targets* (which can be states or *parameters*) are estimated, it is not always obvious from which target a measurement originates. This problem is often denoted as *data association*. For the tracking of multiple, *time-varying* targets, the data association problem is a special case of the DBN in figure 5.4(b), in which the continuous part of the state vector consists of several *independent* targets.

Data association problems typically occur in multi-target tracking (both with a fixed or unknown number of targets) (Reid 1979; Fortmann, Bar-Shalom, and Scheffe 1983; Bar-Shalom and Fortmann 1988; Schulz, Burgard, and Fox 2003; Schulz and Burgard 2001; Vermaak, Godsill, and Pérez 2005), in localization (both (global) localization with a map and Simultaneous Localization And Mapping (SLAM) problems), Simultaneous Localization And People tracking (SLAP) problems with *unknown data association*,[11] and simultaneous Contact Formation recognition and geometrical parameter estimation, dealt with in detail in Chapter 6. In computer vision research, the term *motion correspondence* (Cox 1993) is often used. In all these applications, each measurement has to be linked to a certain "cause". In case of localization, the map consists of several *parameter* features, often denoted as *beacons*. Each measurement is caused by exactly one beacon, by clutter or a non-modeled beacon.[12] For multi-target tracking and SLAP, the *state* vector

[11] Although these applications all have a data association part, not all of them belong to the current Section, see table 5.1.

[12] We can always define a "measurement" such that this statement holds. E.g. if a laser-scanner is used as sensor for localisation, typically each raw laserscan "measurement" will be processed first (e.g. by clustering or line matching techniques, this is most often denoted as *feature extraction*), which results in several feature/beacons "measurements". The former raw "measurement" (the whole laserscan) is not caused by *one* beacon, but the latter "measurements" is. See e.g. (Thrun 1998a; Aycard, Mari, and Washington 2004).

consists of multiple dynamic *targets*. Data association is the process of *linking* the measurements to the beacons or targets. However, under uncertainty, this process is not trivial and the data association is to be considered a Random Variable.

Unlike the algorithms described in the previous Section, most algorithms for data association do not use a system model $P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d)$ describing the evolution of the data association Random Variable. Instead Maximum Likelihood methods are used to estimate their values at each timestep from scratch, to alleviate the computational resources necessary during online estimation and due to the fact that in most cases, one is not explicitly interested in the value of the data association variable.[13] Indeed, consider the case of multi-target tracking with a fixed and known number of targets. This application can be represented by the DBNs of figure 5.5. As shown in figure 5.5(b), each measurement is caused by exactly one target: given the value of the data association variable, the different positions and orientations of the targets can be considered independent of each other:

$$P(\boldsymbol{x}_k^c|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}^d) = \prod_i P(\boldsymbol{x}_k^{c,i}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}^d). \tag{5.5}$$

This means that, if one assumes the data association problem is solved, separate *lower* dimensional filters can be run for each of the targets, yielding a superior performance.

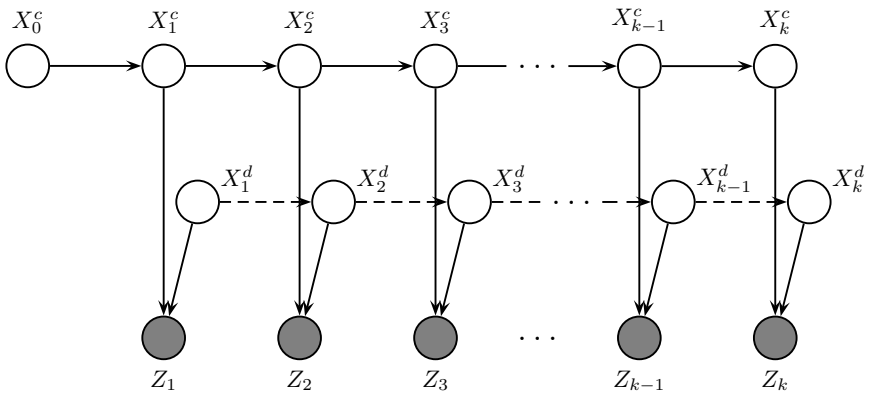
For the above reason, fully Bayesian methods are rarely used to solve the data association problem.[14, 15] Instead, an enormous range of Maximum Likelihood methods is available, most of them based on the Mahalanobis distance (Duda and Hart 1973) between the predicted and the true measurement. *Nearest Neighbour* (NN) filters (Bar-Shalom and Li 1993; Neira and Tardós 2001) choose an association based on the smallest Mahalanobis distance, mostly combined with a gating procedure to reduce computational complexity. Neira and Tardós (2001) discuss a 1D example, in which the different assumptions of these filters and the consequences on the convergence properties of the filter are explained. The *Individual Compatibility Nearest Neighbour* (ICNN) Filter considers each measurement separately and thereby omits correlation information present in related measurements. Since the filter only uses the ML estimate, a wrong association can lead to divergence of the filter taking care of the targets estimation.[16] The sequential versions of

[13] Contrary to what is discussed in Section 5.5.2 and in the next Chapter.

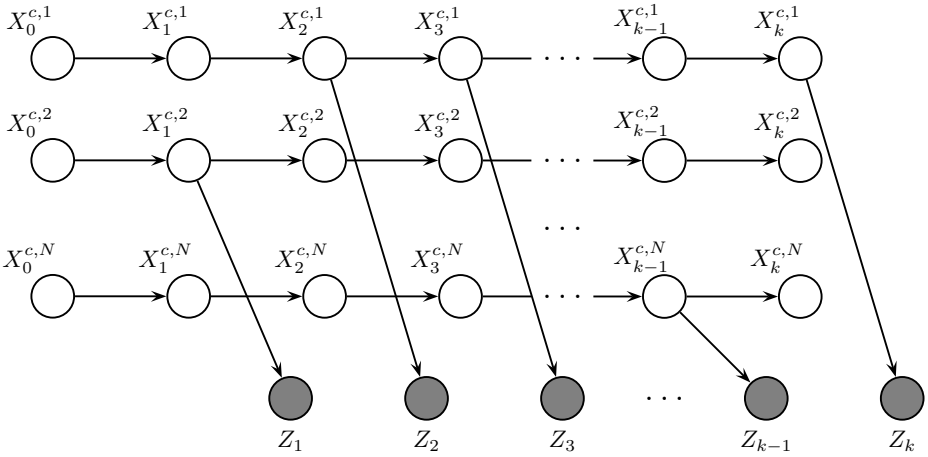
[14] The case dealt with in the next Chapter is a notable exception!

[15] Note however that this ML data association is the main cause for the "closing the loop" divergence in SLAM problems. Other reasons include computational simplifications such as a sparsification of the covariance or information matrices. See e.g. (Paskin 2003) for a discussion on the matter.

[16] Note also that the calculation of the Mahalanobis distance relies on the linearisation of



(a) DBN *with* explicit Data Association Variable. $\boldsymbol{X}^c$ denotes the combined vector of all continuous dynamic target positions, $\boldsymbol{X}^d$ denotes the discrete data association variable, indicating from which target the measurement originates. The dashed arrow represents the optional causal system model $P(\boldsymbol{x}_k^d|\boldsymbol{x}_{k-1}^d)$.



(b) DBN *without* explicit Data Association Variable. $\boldsymbol{X}^{c,i}$ denotes the vector of the ith continuous dynamic target positions. Every measurement is caused by exactly one target.

FIGURE 5.5: Possible DBN representations of multi-target tracking, assuming a fixed number of targets.

the NN algorithm (e.g. the *Sequential Compatibility NN* (Neira and Tardós 2001), *Track Splitting Filter* (Smith and Buechler 1975)) take this information into account, but the order of the measurements influences the result. The *Joint Compatibility NN* filter (branch and bound) (Neira and Tardós 2001) calculates the maximum over all possible orderings of the measurements and applies some optimizations to allow realtime execution. The *Multiple Hypothesis* Filter (Reid 1979; Cox and Leonard 1991; Cox 1993) adds the capability of initiating and terminating (new) features. Unfortunately, the latter algorithm has an exponential complexity and can therefore typically not be executed in realtime. Fortmann, Bar-Shalom, and Scheffe (1983) proposed a suboptimal algorithm, the *Joint Probability Data Association Filter* (JPDAF), which does not suffer from the exponential complexity. This filter calculates a probability distribution over the data association variable at each timestep, without considering its history. Schulz, Burgard, and Fox (2003) and Vermaak, Godsill, and Pérez (2005) implement Particle Filter variants generalizing some of the assumptions made in the original algorithm. Montemerlo, Whittaker, and Thrun (2002) apply a ML approach for simultaneous localization and people tracking, exploiting the conditional independence of the different people, given the robot's location, using a Conditional Particle Filter and Nearest Neighbour data association. Each particle representing the robot's state, has N independent small state size Particle Filters associated with it to track the different people.

On the fully Bayesian data association front, Shumway and Stoffer (1992) mention the possibility to model the data association vector as a discrete truly time-varying variable with Markov history. If the time delay between measurement updates is relatively small compared to the the motion of the targets, $\boldsymbol{x}_k^d$ will be similar to $\boldsymbol{x}_{k-1}^d$. For the case of discrete states and unknown continuous *parameters* (Section 5.5.2), the discrete state *is* the data association value. This thesis presents an example where a system model for the data association is considered, is the simultaneous CF recognition and geometrical parameter estimation described in Section 5.5.2 and the next Chapter.

Note that, for realistic applications, the number of targets/beacons is unknown, adding a discrete state variable to the estimation problem. This results in a varying state size. In order to perform realtime estimation, a maximum number of targets/states should be fixed beforehand, in order to avoid dynamic memory allocation.

"Continuous" data association

All the above (and to the best of my knowledge all research literature about data association) deals with what I refer to as "discrete" data association:

the measurement equation and the assumption of additive Gaussian uncertainty.

A certain measurement is supposed to be caused by exactly *one* feature or beacon (out of a possible series).

So, in the case of robotic localization, this results in a measurement model $P(\boldsymbol{z}_k|\boldsymbol{x}_k^c, [\boldsymbol{\theta}], x_k^d)$. Given the continuous location of the robot $\boldsymbol{x}_k^c$, and the discrete data association state x_k^d denoting from which beacon the measurement stems, we can predict the measurement. In case of an unknown map, $\boldsymbol{\theta}$ represents the unknown beacon locations in the map. In case of the cube-in-corner assembly task of Chapter 6, one obtains a measurement model $P(\boldsymbol{z}_k|\boldsymbol{\theta}, x_k^d)$. $\boldsymbol{\theta}$ represents the geometrical parameter vector, and x_k^d represents the CF at timestep k.

As already mentioned above, often "raw" measurements are processed first (by Bayesian or non-Bayesian techniques) in order to obtain "feature" measurements. Indeed, e.g. a mobile robot localizing itself with a laser scanner using the walls as beacons, will first process the raw laser scan. This will result in a certain number of line (or plane) feature measurements, which are then matched with the features in the current map. For the cube-in-corner assembly task dealt with in Chapter 6, the raw measurement is e.g. *one* position measurement from the robot's encoders. For e.g. a vertex-face Contact Formation this results in an *implicit* measurement equation that states that the vertex with coordinates $[x\ y\ z]^T$ is positioned in the plane with parameters $[a\ b\ c\ d]^T$:

$$ax + by + cz - d = 0. \tag{5.6}$$

Actually, this equation is *implicit* because we don't know *which* point of the plane generated the position measurement. One way to deal with this is to model this equation by means of a *continuous* data association variable $\boldsymbol{x}_k^c$, which (for the cube-in-corner assembly problem) represents the point of the plane that generated the measurement z_k:

$$P(\boldsymbol{z}_k|\boldsymbol{\theta}, \boldsymbol{x}_k^c), \tag{5.7}$$

resulting in an *explicit* measurement equation *if* the continuous data association variable is known. The rigorous Bayesian way of dealing with this unknown continuous data association variable is obtained through marginalization:

$$P(\boldsymbol{z}_k|\boldsymbol{\theta}) = \int P(\boldsymbol{z}_k|\boldsymbol{\theta}, \boldsymbol{x}_k^c) P(\boldsymbol{x}_k^c|\boldsymbol{\theta}) d\boldsymbol{x}_k^c. \tag{5.8}$$

Unfortunately this integration is often too costly to be executed in realtime. Gadeyne and Bruyninckx (2001) calculate off-line *likelihood tables* and use these tables for solving an online global localization problem with a robot arm. An example of such a likelihood table is illustrated in figure 5.6. Note that both problems from figure 5.6 also contain a *discrete* data association problem: Indeed, we suppose we don't know which plane or line caused the measurement.

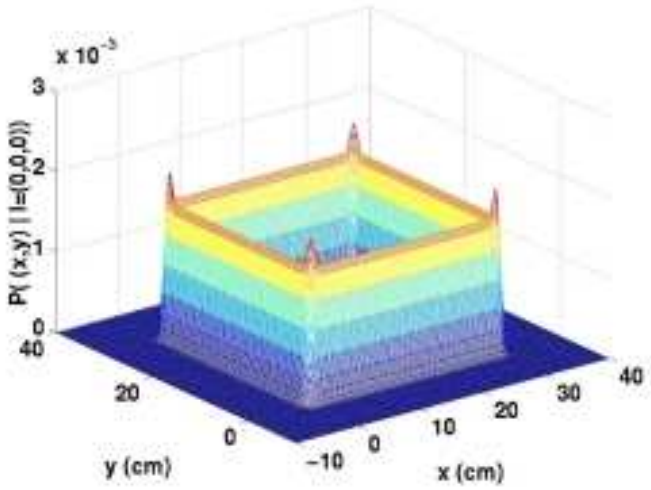


Figure 5.6: Likelihood table for 2D localisation of a cube. The robot localizes the cube by "palpating" the object as illustrated in figure 1.4 on page 7. The measurements for this experiment are the x and y position of the end effector each time the force sensor of the robot exceeds a threshold indicating a contact with the object. The off-line calculation of the table takes a long time, online evaluation of measurement probabilities is very fast. The figure represents the probability of measuring a value $(x\ y)$ if the cube is located at $(0, 0)$ with one of its corners.

If we would add a sensor to the setup of figure 1.1 that provides *global information* about the setup such as a laser scanner or a camera, this would also provide explicit and valuable information. However, such sensors are not always usable due to hazardous environments.

5.4 Pure Parameter Estimation

The first column of table 5.1 refers to situations in which all state RVs are assumed to be known. Models of this type can be described by the (static) BN of figure 5.7.

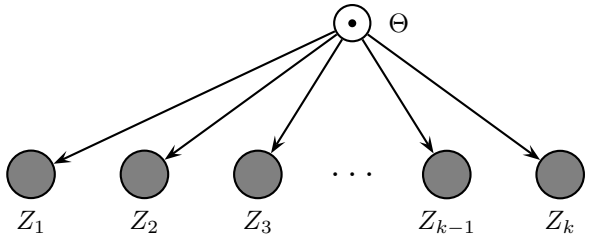


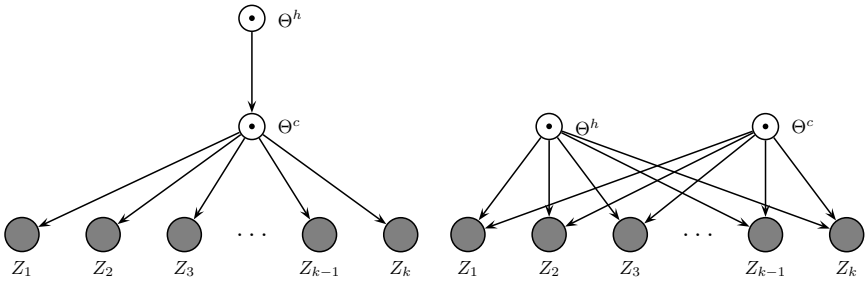
FIGURE 5.7: BN representing a pure parameter estimation problem.

In the case of unknown hyperparameters,[17] it is often useful to split up the parameter vector into parts. Two often found possibilities are shown in figure 5.8. The first case (figure 5.8(a)) is typically used when the influence of the prior density on the posterior is large. The second case (figure 5.8(b)) occurs when there are uncertainties in the model that are not obvious to model, e.g. when the uncertainty on the sensor measurements is unknown and strongly influences the posterior PDF. Depending on which model is chosen, several optimizations in terms of filtering are possible, e.g. in the case of estimation with Sequential Monte Carlo methods and the BN of figure 5.8(b), separate filters could be used for each of the Particle Filters reducing the computational complexity.

5.4.1 Known states, continuous parameters

A typical application of this type of estimation problems is the estimation of geometrical parameters assuming the Contact Formation is known. This problem was dealt with in previous Autonomous Compliant Motion research

[17] Hyperparameters are typically continuous valued!



(a) Causal relationship between hyperparameters.

(b) Unknown model uncertainties.

Figure 5.8: Parameter estimation problems involving hyperparameters $\boldsymbol{\theta}^h$.

at PMA (Lefebvre 2003). As mentioned in Chapter 2, *recursive* inference (and thus filtering) for this type of problems is important in order to adapt online the controller algorithms as the information about the geometrical parameters increases, and to perform *active sensing*.

All analytical filters described in Section 5.3.1 can be used to estimate $P(\boldsymbol{\theta}|\boldsymbol{z}_{1:k})$. In this case, the NMSKF is extremely valuable—especially once the posterior has converged to a Gaussian—since model approximation errors do not accumulate for this type of filter, and it is always applicable to pure parameter estimation.

As already mentioned in the previous Chapter, basic Particle Filter algorithms have trouble dealing with (pure) parameter estimation.[18] Indeed, in case there are only parameters, Bayesian inference reduces to

$$P(\boldsymbol{\theta}|\boldsymbol{z}_{1:k}) \sim P(\boldsymbol{z}_k|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{z}_{1:k-1}). \tag{5.9}$$

This means that, for parameter estimation, there is no *recursive* optimal proposal distribution and only the particle *weights* are updated when filtering, leading to a very small support for the estimation of posterior properties. Furthermore, Section 4.5 explained that the series c_k (eq. (4.20)) is monotonically increasing in the case of parameter estimation. So, to ensure the Monte Carlo variance of the estimator stays within a given bound, more samples are necessary if t increases.

[18] Note that these statements and the following paragraphs concern *all* problems involving parameter estimation of *any* type.

The problem of *recursive parameter estimation* with sequential Monte Carlo methods only recently got attention from the research community. The application of an *MCMC*-step (Section 4.4.2) or the use of methods based on *rejection sampling* (Section 4.4.3) solve the problem of the fixed support, but mostly these methods are applied at the cost of losing the *realtime* aspect of estimation.

Liu and West (2001) propose the use of *kernel density estimation methods* (Silverman 1986; West 1993a; West 1993b). At each timestep, an analytic kernel (most often a unimodal or multimodal Gaussian) is fitted through the current sample points. New samples are generated from this analytic kernel and used for the weights update of the following filter step. This is sometimes referred to as the *Regularized Particle Filter* (RPF) (Musso, Oudjane, and LeGland 2001).

The introduction of *artificial dynamics* on the parameters (Acklam 1996; Kitagawa 1998; Liu and West 2001) also alleviates the problem of the small support.[19] In this case, the parameter vector $\boldsymbol{\theta}$ is considered as a state and typically zero mean Gaussian uncertainty is added:[20]

$$P(\boldsymbol{\theta}_{k+1}|\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{\theta}_k, \boldsymbol{\Sigma}_{\theta,k}). \tag{5.10}$$

Of course, the artificial dynamic leads to a *loss of information*. Liu and West (2001) propose to introduce an artificial *correlation* between $\boldsymbol{\theta}_k$ and the zero mean additive Gaussian uncertainty. Indeed, for state space models with additive uncertainty

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{f}(\boldsymbol{\theta}_{k-1}) + \boldsymbol{\rho}_{k-1,s}, \\ \boldsymbol{z}_k &= \boldsymbol{h}(\boldsymbol{\theta}_k) + \boldsymbol{\rho}_{k-1,m}, \end{aligned} \tag{5.11}$$

where $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ denote system measurement model, respectively, the system measurement additive uncertainty sequences $\boldsymbol{\rho}_{k-1,s}, \boldsymbol{\rho}_{k-1,s}$ are usually assumed independent of each other and of the state vector $\boldsymbol{\theta}$.[21] In the case of the introduction of artificial dynamics, eq. (5.11) reduces to

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{\theta}_{k-1} + \boldsymbol{\rho}_{k-1,s} \\ \boldsymbol{z}_k &= \boldsymbol{h}(\boldsymbol{\theta}_k) + \boldsymbol{\rho}_{k-1,m}, \end{aligned} \tag{5.12}$$

where $\boldsymbol{\rho}_{k-1,s} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\theta,k})$. By introducing correlation between $\boldsymbol{\theta}$ and $\boldsymbol{\rho}_{k-1,s}$, we can assure that the covariance is not altered during the system update.

[19] A similar ad hoc implementation was already introduced in (Gordon, Salmond, and Smith 1993) and coined as *jittering*.

[20] Note that the basic procedure described hereafter (not taking into account correlation) can also be considered as a kernel density estimation method, in which the analytic kernel is a multimodal Gaussian with a mode for each particle, and exactly one sample is generated from each mode.

[21] This assumption is mostly valid given the physical model.

Indeed, given the linear system eq. (5.12)

$$Var_{P(\boldsymbol{\theta}_k|\boldsymbol{z}_{1:k})}[\boldsymbol{\theta}_k] = Var_{P(\boldsymbol{\theta}_{k-1}|\boldsymbol{z}_{1:k-1})}[\boldsymbol{\theta}_{k-1}] + \boldsymbol{\Sigma}_{\theta,k-1} + 2C(\boldsymbol{\theta}_{k-1}, \boldsymbol{\rho}_{k-1,s}) \tag{5.13}$$

where C denotes the correlation between $\boldsymbol{\theta}, \boldsymbol{\rho}_{k-1,s}$. If the sum of the two last terms is zero, the covariance matrix is not altered by the introduction of the artificial dynamics.[22] Usually, the covariance is also reduced as more measurements become available:

$$\boldsymbol{\Sigma}_{\theta,k} = \alpha \boldsymbol{V}_k, \tag{5.14}$$

where V_k is a Monte Carlo estimate of $Var_{P(\boldsymbol{\theta}_k|z_{1:k})}[\boldsymbol{\theta}_k]$ and $\alpha < 1$.

Note that, contrary to the above described methods which are fully Bayesian, for pure parameter estimation problems, (recursive) Maximum Likelihood methods can be applied that estimate the maximum of the likelihood function

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}) = \prod_{i=1}^{k} P(\boldsymbol{z}_i|\boldsymbol{\theta}), \tag{5.15}$$

where the last equality is only valid if the Markov assumption holds and the measurements are i.i.d. The likelihood is a function (not a PDF) of the unknown parameter $\boldsymbol{\theta}$, and a PDF over the measurements $\boldsymbol{z}_i$. These methods often use gradient techniques to search and a detailed treatment is out of the scope of this Chapter.

This thesis uses the introduction of artificial dynamics for the geometrical parameter estimation experiment described in the next Chapter, as the application of a MCMC step would be computationally prohibitive.

5.4.2 Known states, discrete parameters

In the case of fully discrete parameters, the posterior can easily be updated using Bayes' rule. This is often called *hypothesis testing*. Most Bayesian textbooks (e.g. (Jaynes 1996)) deal extensively with this case.

5.4.3 Known states, hybrid parameters

Just as in the case of pure hybrid states, there's often a causal relationship between the discrete and the continuous part of $\boldsymbol{\theta}$. This is visualized by the BN of figure 5.9. A typical application of this type is the construction of a 3-D geometrical model from sensor information during autonomous compliant motion (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004; Slaets, Rutgeerts,

[22] Note however that, in the case of parameter estimation, as the number of measurements grows, the uncertainty further reduces and even analytical approaches often add extra uncertainty to the parameter to avoid numerical problems.

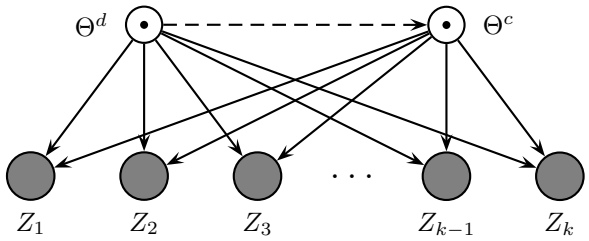


FIGURE 5.9: A particular case of pure hybrid parameter estimation. Each hypothesis (an instantiation of the discrete RV $\boldsymbol{\theta}^d$) comes with its own measurement model $P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^d, \boldsymbol{\theta}^c)$ describing the relation between the measurements $\boldsymbol{z}_{1:k}$ and the continuous part of the parameter vector $\boldsymbol{\theta}^d$. The dashed arrow denotes an optional causal prior $P(\boldsymbol{\theta}_c|\boldsymbol{\theta}_d)$.

Gadeyne, Lefebvre, Bruyninckx, and De Schutter 2004). These papers use sensor measurements to build a 3-D model from a limited set of geometrical primitives.

A cube is inserted into a corner with a serial manipulator (see Chapter 6). During this assembly task, wrench and position measurements are used to detect certain Contact Formations and estimate geometrical parameters. Each time a new contact is detected (by a SNIS test, see Section 5.3.3), a number of NMSKFs (Lefebvre, Bruyninckx, and De Schutter 2005a) is started to estimate the set of geometrical parameters, each with a different measurement model corresponding to one given hypothesis. The probability of a certain hypothesis is calculated *recursively* as follows:[23]

$$\begin{aligned} P(\boldsymbol{\theta}^d|\boldsymbol{z}_{1:k}) &= P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k-1})P(\boldsymbol{\theta}^d|\boldsymbol{z}_{1:k-1}) \\ &= \int P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{\theta}^c)P(\boldsymbol{\theta}_c|\boldsymbol{\theta}_d, \boldsymbol{z}_{1:k-1})d\boldsymbol{\theta}^c P(\boldsymbol{\theta}^d|\boldsymbol{z}_{1:k-1}) \end{aligned} \tag{5.16}$$

where $\boldsymbol{\theta}^d$ represents the discrete hypothesis variable (each CF has a different set of hypotheses), and $\boldsymbol{\theta}^c$ represents the geometrical parameter vector. Note that, in the first line $P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k-1}) \neq P(\boldsymbol{z}_k|\boldsymbol{\theta}^d)$ although the system is Markovian. Indeed, only knowledge about the discrete model parameter *and* the continuous parameter suffices to omit the previous measurements!

Assuming the posterior estimate of the NMSKF is unimodal Gaussian *and*

[23] Of course, this derivation applies not just to this example but to all hybrid parameter systems dealt with in this Section!

it has converged[24], the integral can be calculated in a recursive way analytically.

More generally, N analytic filters in parallel can be used to estimate $P(\boldsymbol{\theta}^c|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k})$, but the integral (5.16) will seldom be analytically tractable. Moreover, assuming a unimodal Gaussian is used to approximate $P(\boldsymbol{\theta}^c|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k})$, this means that the hypothesis test only generates valid results once enough measurements are taken. Unfortunately, the posterior probability of the model is calculated in a recursive manner, and for large uncertainties, the assumptions typically don't hold. Nevertheless, Slaets et al. (2004) obtain good results using this approach. Note that, taking into account the considerations of Section 5.4.1, a single Particle Filter could be used to track the full hybrid posterior $P(\boldsymbol{\theta}^c|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k})$. (Lee and Chia 2002) use a Particle Filter extended with a MCMC step (using a Metropolis-Hastings kernel) to perform simultaneous parameter estimation and static model selection in signal processing applications.

Sometimes, to alleviate computational requirements, the term

$$P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k-1}) = \int P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{\theta}^c) P(\boldsymbol{\theta}_c|\boldsymbol{\theta}_d, \boldsymbol{z}_{1:k-1}) d\boldsymbol{\theta}^c \tag{5.17}$$

is replaced by a Maximum likelihood estimate:

$$P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k-1}) \approx P(\boldsymbol{z}_k|\boldsymbol{\theta}^d, \boldsymbol{\theta}^{c,*}). \tag{5.18}$$

Unfortunately this approximation leads to *overfitting*, a phenomenon which is also encountered in Neural Networks (also using a point estimate of parameters). An enormous number of (Bayesian and non-Bayesian) techniques have been proposed to deal with the overfitting problem, amongst others methods using the *Akaike Information Criterion* (AIC) (Akaike 1973), the *Bayesian Information Criterion* (BIC) (Schwarz 1978), ...[25] MacKay (1995) describes how Neural Network learning can be seen as Bayesian learning and how the regularization parameters to avoid overfitting can be calculated in a Bayesian way.

Pairwise comparison. In case there are only two hypotheses $\boldsymbol{\theta}^{d,1}$ and $\boldsymbol{\theta}^{d,1}$ or hypotheses are compared pairwise, the ratio

$$\frac{P(\boldsymbol{\theta}^{d,1}|\boldsymbol{z}_{1:k})}{P(\boldsymbol{\theta}^{d,2}|\boldsymbol{z}_{1:k})} \tag{5.19}$$

[24] i.e. the KL-distance between $P(\boldsymbol{\theta}^c|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k})$ and $P(\boldsymbol{\theta}^c|\boldsymbol{\theta}^d, \boldsymbol{z}_{1:k-1})$ is small.

[25] See e.g. (Lefebvre 2003) p. 42–44 for an overview.

is called the *posterior odds*. If prior information is not considered (or a uniform prior over both hypotheses is assumed), the ratio

$$\frac{P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{d,1})}{P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{d,2})} \tag{5.20}$$

is the same as the posterior odds (via Bayes' rule); it is called the *likelihood ratio* or *Bayes' factor* (Kass and Raftery 1995).

Occam's Razor and overfitting. The previous model selection algorithms are all fully Bayesian and *therefore incorporate* the principle of Occam's razor (Jeffreys 1939). For 1D, the principle is shown in figure 5.10.

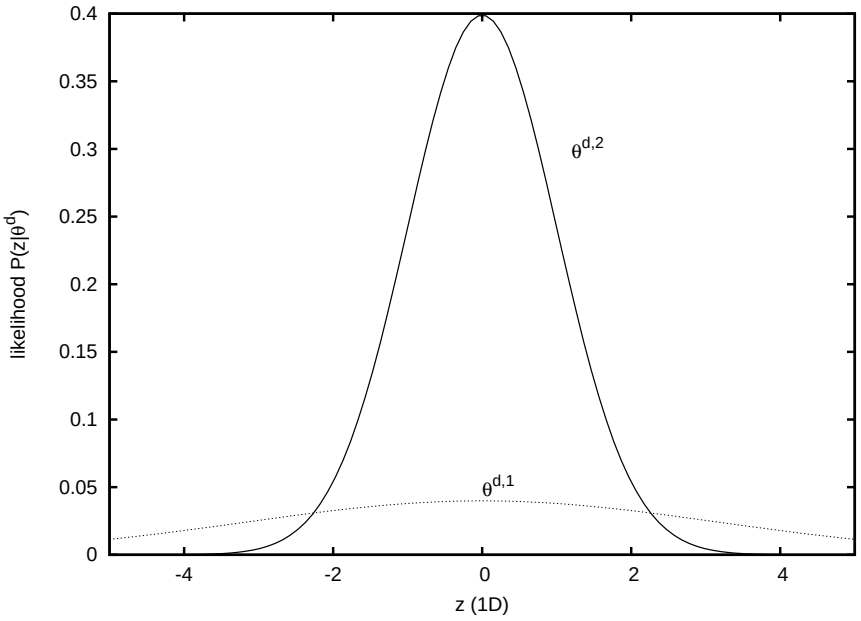


FIGURE 5.10: 1D Illustration of Occam's razor principle. Model 1 corresponds to a CF with a small covariance matrix, model 2 to a model with a larger covariance matrix, capable of explaining more measurements. If the measurement's value is around 0, both models explain it, but model 1 will be preferred.

5.5 Combined state and parameter estimation

The remaining 9 cells in the bottom-right part of table 5.1 are solved with a combination of methods discussed in the previous Sections. These problems are often denoted as *incomplete data problems*.[26] The variety of theoretically possible BN models increases if both parameters and states are involved, but in practice, only a limited number of models is used. The most general case is represented by figure 5.11, but the following sections will discuss some more specific BN architectures.

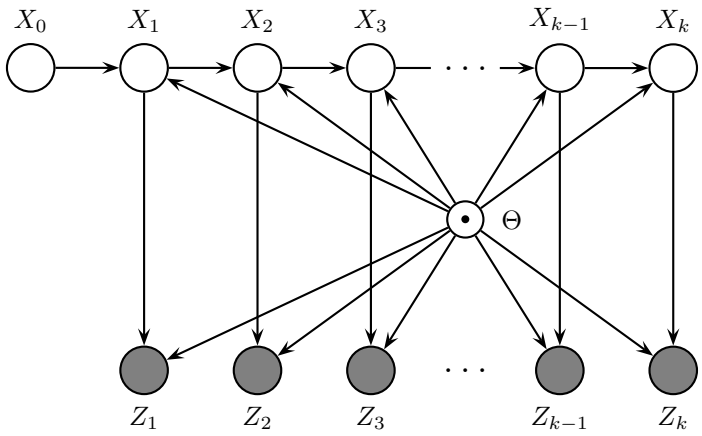


Figure 5.11: Bayesian network representing the joint posterior in its most general form (compare with eq. (2.6)). The dotted circle (Θ) denotes a parameter vector (i.e. all hidden RVs that do not change over time in the particular (BN) model used), empty circles denote states (i.e. hidden RVs that change over time according to a certain *system* model) and grey circles denote observed variables such as measurements.

The most "evident" choice for combined state/parameter estimation is to tie states and parameters together into one Random Variable and use a filter to perform estimation for this new variable. This principle is referred to as *data augmentation* (Tanner 1992; Tanner 1996). However, if the dimension of the parameter vector is large, the resulting state estimation problem is sometimes hard to solve under realtime constraints. In those cases, the computationally less expensive but not fully Bayesian *Expectation-Maximization*

[26] The term "incomplete data problems" originates from research areas dealing mainly with pure state estimation problems, and the addition of unknown parameters is referred to as incomplete data.

algorithm (EM) (Dempster, Laird, and Rubin 1977) is often used as an alternative to estimate $\boldsymbol{\theta}$ off-line first. Typically, the term *training* is used for such an off-line parameter learning process. The obtained values for the parameters can be used in a state estimation filter afterwards (Section 5.3).

5.5.1 Continuous states, continuous parameters

Although the principles of data augmentation and the EM-algorithm are discussed in detail for continuous systems in this Subsection, they apply[27] to all combined state and parameter estimation problems.

Data augmentation

Data augmentation is the term used for augmenting the state vector with unknown parameters and performing full Bayesian inference over the resulting RV.[28] Indeed, from a Bayesian information-technical point of view, there is no fundamental reason treating parameters in a different way than states. In the case of the Kalman Filters, this algorithm is sometimes referred to as *Joint Kalman Filtering* (Wan and Nelson 1997). If sequential Monte Carlo methods are used to estimate the joint posterior, the remarks concerning parameter estimation with Particle Filters of Section 5.4.1 for the parameter part of the hybrid random variable apply.

SLAM with known data association is a typical application of data augmentation: the unknown state is the robot's time-varying position, and the static map is represented by unknown parameters. Kalman Filter variants are very popular to solve this problem (Dissanayake, Newman, Clark, Durrant-Whyte, and Csorba 2001). If too many landmarks are used (e.g. in large environments), updating the Kalman Filter in realtime becomes too resource intensive. However, in the case of SLAM, the information matrix is typically sparse and the *Information Filter* is much faster than the standard (IE)KF (Thrun, Liu, Koller, Ng, Ghahramani, and Durrant-Whyte 2004b).

The *FastSLAM* algorithm (Montemerlo, Thrun, Koller, and Wegbreit 2003; Montemerlo, Thrun, Koller, and Wegbreit 2002) is a Rao-Blackwellized Particle Filter (Murphy and Russell 2001) that exploits an independence relationship of the joint posterior. Indeed, given the position $\boldsymbol{x}_k$ of the robot, the positions $\boldsymbol{\theta}^i$ of the N individual landmarks are independent of each other:

$$P(\boldsymbol{\theta}^{1:N}, \boldsymbol{x}_k | \boldsymbol{z}_{1:k}) = \prod_{i=1}^{N} P(\boldsymbol{\theta}^i | \boldsymbol{x}_k, \boldsymbol{z}_{1:k}) P(\boldsymbol{x}_k | \boldsymbol{z}_{1:k}). \tag{5.21}$$

[27] Note that approximations are sometimes necessary.

[28] Some authors also use the term *self-organizing* (state-space) model to denote this approach (Kitagawa 1998).

A Particle Filter is used to estimate the robot's location and each particle has N EKFs associated with it to estimate the position of the N landmarks.

The Expectation-Maximization algorithm

The Expectation-Maximization algorithm (Dempster, Laird, and Rubin 1977; Shumway and Stoffer 1982) subdivides the estimation problem into two steps: a state estimation step and a parameter learning step. It is an off-line, iterative *Maximum Likelihood* method for searching a *local*[29] maximum over $\boldsymbol{\theta}$ of the PDF $P(\boldsymbol{z}_k|\boldsymbol{\theta})$. The EM-algorithm tries to find the maximum of the *marginal* likelihood function $P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta})$:[30]

$$\boldsymbol{\theta}_{EM} \approx ArgMax_\theta \; P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}) = ArgMax_\theta \int P(\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}|\boldsymbol{\theta}) d\boldsymbol{x}_{1:k}. \tag{5.22}$$

In EM-speak, the latter function is denoted as the *observed data likelihood* function:

$$\mathcal{L}^o(\boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}). \tag{5.23}$$

Using the product rule, we can relate this likelihood with the *complete data likelihood* function $\mathcal{L}^c(\boldsymbol{\theta}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}) = P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})$:

$$P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}) = \frac{P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})}{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}, \boldsymbol{z}_{1:k})} \quad ; \quad \mathcal{L}^o(\boldsymbol{\theta}|\boldsymbol{z}_{1:k}) = \frac{\mathcal{L}^c(\boldsymbol{\theta}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k})}{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}, \boldsymbol{z}_{1:k})}. \tag{5.24}$$

The core idea behind the EM-algorithm is to search the parameter vector $\boldsymbol{\theta}^*$ that maximizes the *complete data likelihood* function, since we can prove (Appendix A) that $\boldsymbol{\theta}^*$ also maximizes the *observed data likelihood* function. $\boldsymbol{\theta}^*$ is calculated iteratively in two steps:

- The E-step estimates the unknown states, by running a filter, *given* the current iteration value $\boldsymbol{\theta}^t$,[31] to estimate $P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})$.
- The M-step maximizes $\boldsymbol{\theta}$, (or increases, but this converges more slowly and is sometimes referred to as *generalized* EM (Dempster, Laird, and Rubin 1977).)

  $$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = E_{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})} \left[log \left(P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta}) \right) \right], \tag{5.25}$$

  using the posterior from the E-step. This maximization requires the calculation of an integral, which can sometimes be solved or approximated analytically, but typically requires Monte Carlo methods (Doucet and Tadic 2003; Andrieu and Doucet 2003).

[29] Variants to avoid getting stuck in local maxima exist, see e.g. (Ueda and Nakano 1998).

[30] Note that this is equivalent to searching the maximum of the marginal a posteriori density $P(\boldsymbol{\theta}|\boldsymbol{z}_{1:k})$, *given a uniform prior!*

[31] The *superscript* ^{t} denotes an particular value of $\boldsymbol{\theta}$ during the iteration process, and has nothing to do with the subscript k that relates to the time of the filter!

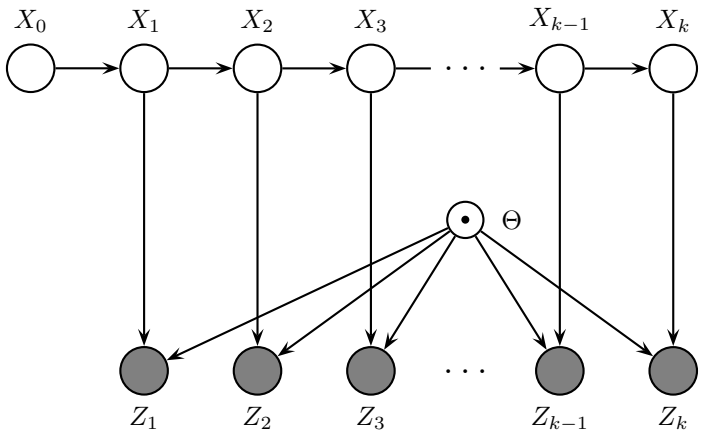
EM is often used for map building purposes (Thrun 2003). Sometimes specific assumptions lead to online execution (Doucet and Tadic 2003; Andrieu and Doucet 2003; Thrun, Martin, Liu, Hähnel, Emery-Montemerlo, Chakrabarti, and Burgard 2004).

5.5.2 Discrete states, continuous parameters

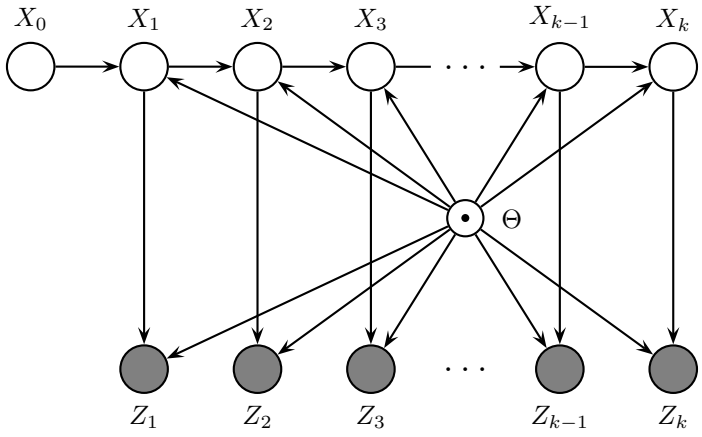
This section discusses three types of networks. The most frequent BN topology is the Hidden Markov Model with unknown model parameters as shown in figure 5.12(a). The best known algorithm for this case is the combination of the Viterbi algorithm—a discrete state version of EM using the forward-backward algorithm (Rabiner 1989) in its E-step—with the *Baum-Welch* algorithm (Baum, Petrie, Soules, and Weiss 1970). After the HMM has been trained with the Baum-Welch algorithm (and thus point estimates of its unknown parameters are available), the states of the HMM can be estimated with the Viterbi algorithm (Section 5.3.2). Note that in this case, people do not use the term *data association* (Section 5.3.4) for the correspondence between the continuous parameter and the state, since there is no online inference process on $\boldsymbol{\theta}$.

A second application of this type of problems is the simultaneous (discrete) Contact Formation recognition and (continuous) geometrical parameter estimation problem dealt with in the next Chapter, visualized by the BN of figure 2.3, where $\boldsymbol{X}$ denotes the discrete Contact Formation state variable (the data association variable) and $\boldsymbol{\theta}$ the geometrical parameter vector. Chapter 6 shows how to perform full online Bayesian inference for this particular case, based on the simplified model of figure 5.12(a). Note that, although the latter BN model is exactly the same model as a Hidden Markov Model with unknown parameters, the combination of a Viterbi algorithm with off-line training methods such as the Baum-Welch algorithm is not suitable for this type of problem. Indeed, the geometrical parameter vector will vary from assembly task to assembly task, yielding *off-line* training with a ML like algorithm useless. This problem is an example of data association where full inference is performed over this state, contrary to the Maximum Likelihood approaches described in Sections 5.3.4 that do not take into account the dynamics of the data association variable. Note that, if the data association problem is merely about deciding if the measurements originate from the assumed model or not (i.e. a *binary* discrete option), a consistency test (such as NIS or SNIS in the case the measurement uncertainties are additive Gaussian) can be used to make this decision.

Another application of continuous-parameter–discrete-state estimation problems is found in data *classification*. Figure 5.13 shows an example and the according BN. Given the fact that (in this case) data are generated from two sources, the goal is to estimate the characteristics of these sources (often

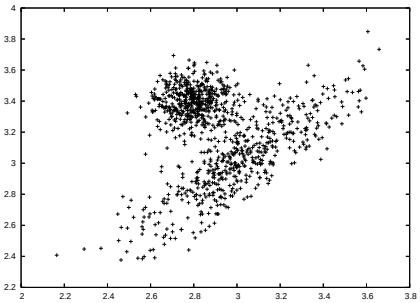


(a) HMM with unknown parameters.

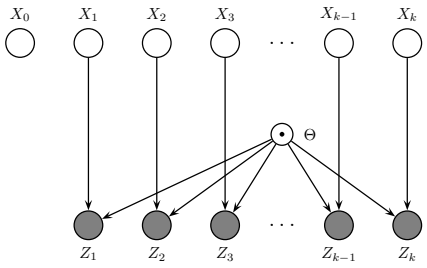


(b) Mixed model where the parameters also influence the evolution of the states. This model will be used for the application in the next Chapter.

Figure 5.12: Possible cases with discrete states and continuous parameters.



(a) Artificial data generated from two 2-dimensional Gaussian PDFs.



(b) BN for data classification.

FIGURE 5.13: Data Classification problem.

represented by continuous hyperparameters) and to assign each data point to a certain target (also a *data association* problem). If the estimation of the parameters happens off-line first and the number of sources generating data is assumed to be known, this process is often referred to as *supervised learning*.[32] The *Bayes' Point Machine* (Watkin 1993) is a Bayesian alternative to deterministic classifiers, in which the data association problem is solved by using a maximum likelihood approach, whereas the posterior density on the unknown parameter vector $\boldsymbol{\theta}$ is also replaced by it maximum likelihood estimate $\boldsymbol{\theta}^*$:

$$P(\boldsymbol{x}^{DA}|\boldsymbol{z}_{1:k}) = \int P(\boldsymbol{x}^{DA}, \boldsymbol{\theta}|\boldsymbol{z}_{1:k}) d\boldsymbol{\theta} \approx P(\boldsymbol{x}^{DA}|\boldsymbol{\theta}^*, \boldsymbol{z}_{1:k}). \tag{5.26}$$

If the number of *data clusters* is unknown too, both the parameters and the number of the densities have to be estimated. This typically happens off-line during a training stage. When the training set data are not labeled (i.e. even for the training set it is unknown which target generated the data), the process is referred to as *unsupervised learning*. After that, data classification is performed using Maximum Likelihood methods.

[32] In the field of data classification, there's also an enormous number of *non*-Bayesian techniques. The off-line learning process for these algorithms is mostly referred to as *training*. (Herbrich 2002) contains an overview of both Bayesian and other classifier algorithms. Note that Bayesian inference can also be used for determining the unknown parameters of Neural Network models or Support Vector Machines (MacKay 1995) and one should make the distinction between mathematical and stochastical models as described in Section 1.4.

5.5.3 Hybrid states, continuous parameters

SLAM with unknown data association, one of the areas on which most recent research in robotics focuses, is an application of these type of models. For solving a full SLAM problem, the following variables are unknown:

- The location of the robot, which is a *continuous state* ($\boldsymbol{x}^c$).
- The (varying) number of beacons ($\boldsymbol{x}^{d,NB}$) and the data association variables ($\boldsymbol{x}^{d,DA}$), which are both *discrete states*.
- The location of the beacons, which are *continuous parameters*.

Mostly, SLAM is modeled as shown in figure 5.14. Several notes apply to this figure:

- Data association is handled using a ML approach. A full Bayesian approach with system model $P(x_k^{d,DA}|x_{k-1}^{d,DA})$ for the data association state is not used in practice, mainly for computational reasons.
- The parameter vector consist of several beacons whose location is independent given the robot's position and the data association variable. Rao-Blackwellized approaches are useful to reduce the computational complexity of the estimation problem.
- The number of features is rarely known. Section 5.5.6 describes how to deal with an unknown number of features.

So, full SLAM is always solved using a mix of Bayesian filters and Maximum Likelihood methods. The problem has been solved with a wide variety of filter algorithms: standard EKFs (Dissanayake, Newman, Clark, Durrant-Whyte, and Csorba 2001), Extended Information Filters (Thrun, Liu, Koller, Ng, Ghahramani, and Durrant-Whyte 2004a), Particle Filters (Fox, Thrun, Burgard, and Dellaert 2001), Rao-Blackwellized Particle Filters (Montemerlo, Thrun, Koller, and Wegbreit 2003). Depending on the specific sensors, beacons and models that are used, some filters are more appropriate than others.

In fact, all the applications described in Section 5.3.3, where not all parameters are known, belong to this category. Mostly these unknown parameters are not estimated using the principle of data augmentation due to computational limitations, but with a variant of EM. Hamilton (1990) describes how an approximation of the EM algorithm can be applied to these models. Sometimes MCMC methods are also used to *train* these unknown model parameters off-line (Murphy 1998).

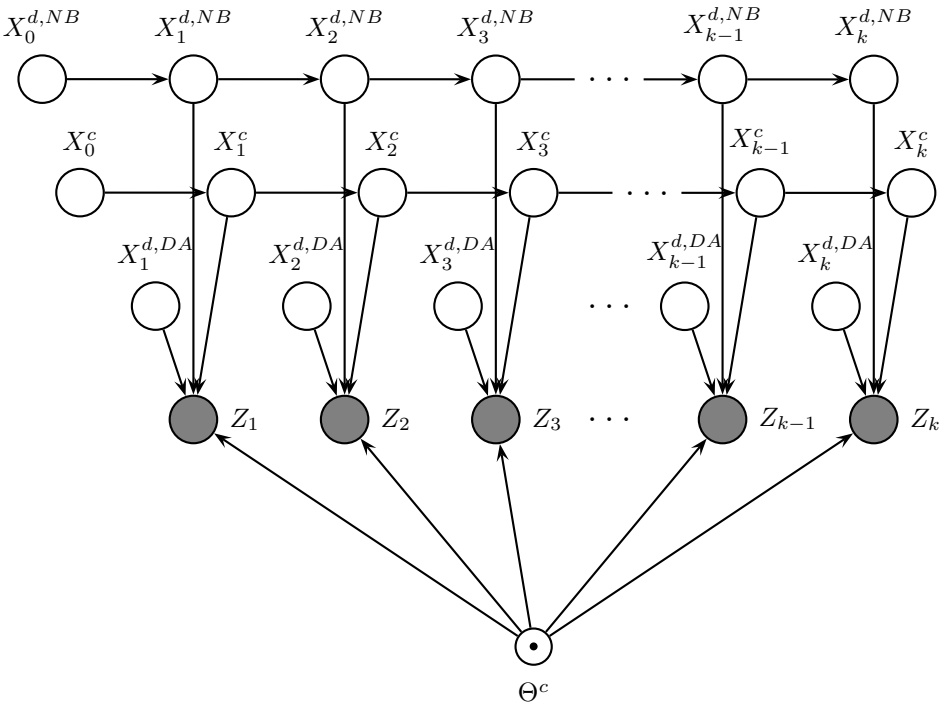


FIGURE 5.14: DBN representing the full SLAM problem with unknown data association and a varying number of features. $X_k^{d,NB}$ represents the number of beacons used for the map at timestep k, X_k^c denotes the position and orientation of the robot, θ^c denotes the position and orientation of the features. $X_k^{d,DA}$ denotes the association vector at timestep k, which is different at each timestep but usually considered as a parameter. Note that the number of beacons is estimated based on the number of "measurement features" detected in the raw measurements. This means that each raw measurement is preprocessed, resulting in a size-varying *vector* of feature measurements Z^k. The data association RV is a discrete vector indicating the association between beacons and measurement features.

5.5.4 Continuous states, discrete parameters

A typical application of this type of system occurs when comparing two (or more) system models with e.g. different distinct continuous hyperparameters, instead of performing full inference over the whole hyperparameter space which might be too costly, or instead of using the EM algorithm which typically cannot be used for online purposes. This situation is shown by the DBN in figure 5.15. From an analytical point of view, as Kalman Filter variants

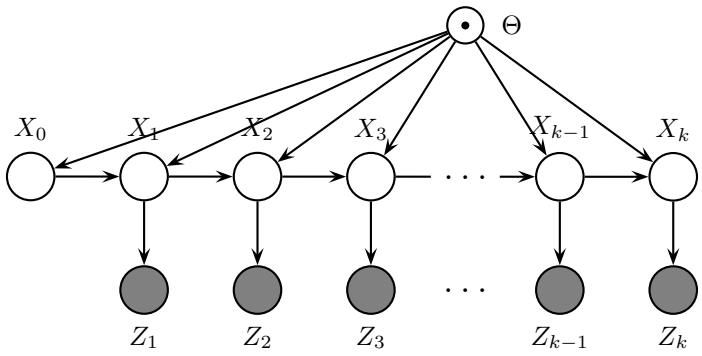


Figure 5.15: Simultaneous state and parameter estimation. If $\boldsymbol{\theta}$ is continuous, this concerns state space models where some parameters of the system model are unknown. If $\boldsymbol{\theta}$ is discrete, this is an application of simultaneous model selection and state estimation.

cannot handle discrete variables, a multiple model approach with a number of filters running in parallel and the calculation of a *Bayes' factor* (Jeffreys 1939; Kass and Raftery 1995) for the discrete model selection as described in (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004) is appropriate. Particle Filters can estimate this hybrid joint posterior density, although the caveats about parameter estimation hold (see Section 5.4).

5.5.5 Discrete states, discrete parameters

A typical example of this type of models is simultaneous speech and speaker recognition. This problem is visualized by the network of figure 5.15. These problems could be tackled online by Particle Filters. The Viterbi algorithm cannot be applied in this case, since it only provides a Maximum Likelihood estimate and cannot be combined with e.g. a Bayes' factor method to perform the model selection problem.

5.5.6 Hybrid states, discrete parameters

In reality, the number of targets in problems such as multi-target tracking of Simultaneous Localization and People Tracking (SLAP), discussed in Section 5.5.4, is rarely known. An extra, discrete state variable allows to estimate the number of targets given the measurements. Schulz, Burgard, and Fox (2003) estimate the evolution of the number of targets, based on the number of features that are detected in the raw measurements. Let the number of features detected in the kth raw measurement be f_k, and t_k the number of targets at timestep k, then

$$P(t_k|f_{1:k}) \sim P(f_k|t_k) \sum_i \left[P(t_k|t_{k-1} = i)P(t_{k-1} = i|f_{1:k-1})\right]. \tag{5.27}$$

Since the inference is only based on the number of detected features, t_k can be estimated by a *separate* Particle Filter or the Viterbi algorithm. The MAP estimate is then used to determine if an extra target is to be introduced or if a target has to be removed. If a target has to be removed, the target with the smallest sum of particle weights (before normalizing) is removed. The discrete measurement model $P(f_k|t_k)$ specifies how many features are visible given a number of targets, and is constructed based on training data taking into account occlusion. The system model $P(t_k|t_{k-1} = i)$ is modeled as a Poisson process.

5.5.7 Discrete states, hybrid parameters

A typical application of these models is to simultaneously distinguish between different *unmodeled* speakers and recognize what they say. If the number of speakers is also unknown, there's an extra discrete state that has to be estimated. Typically for such applications, full online inference is not achievable in realtime and a Baum-Welch algorithm will be used for training purposes first.

5.6 Conclusions

This Chapter provided a classification of Bayesian algorithms and applications, based on the parameters and states being continuous, discrete or hybrid. This classification is complementary in nature to a classification of Bayesian filters according to their representation of the joint posterior density and it helps to situate where the Maximum Likelihood and Maximum A Posteriori algorithms are useful in reducing the computational complexity of full Bayesian inference in real-life situations.

Knowing in which category a certain model belongs is an important step in choosing the most appropriate algorithm for inference, although typically

extra, problem specific, information is necessary to make a final decision. E.g. performing SLAM with sonar sensors using natural beacons in a large environment will require another algorithm than SLAM in a small office building using a laserscanner.

The Chapter also argued that estimation research currently only focusses on a *subset* of available models and describes some new graphical models previously unused in estimation. The next Chapter thoroughly discusses one major experimental case of table 5.1: Rigorous Bayesian Simultaneous Contact Formation recognition and geometrical parameter estimation, which uses the Bayesian Network model involving cross dependency between states and parameters described in Section 5.5.2.

This Chapter also described the link between the BN models and the models used in literature for solving discrete *data association* problems and shows how implicit measurement equations can be considered as *continuous data association* problems. The design of BFL, the C++ estimation library developed in this thesis (Chapter 7) is also based on this classification.

Chapter 6

Hybrid Model-Parameter Estimation applied to Simultaneous Contact Formation Recognition and Geometrical Parameter Estimation

6.1 Introduction

This Chapter presents a novel approach to increase the robot's autonomy during a "one-off" compliant task in a *poorly structured environment*. The robot must find out the geometry of its force-controlled interaction with the environment, for example during deburring of cast pieces without expensive fixtures, assembly, maintenance and autonomous manipulation skills in teleoperation, etc. *Compliant motion* tasks (De Schutter and Van Brussel 1988) are tasks in which a robot manipulates a tool or work piece in contact with the environment. In industrial environments, compliant motion tasks are often position-controlled and hence require *structured environments* (i.e., the work pieces are accurately positioned and their dimensions are known), and/or *(instrumented) passive compliances*. Figures 6.1 and 6.2 show the execution of the particular autonomous compliant motion task whose experimental data is illustrated later in this Chapter: a robot manipulator, equipped with a force

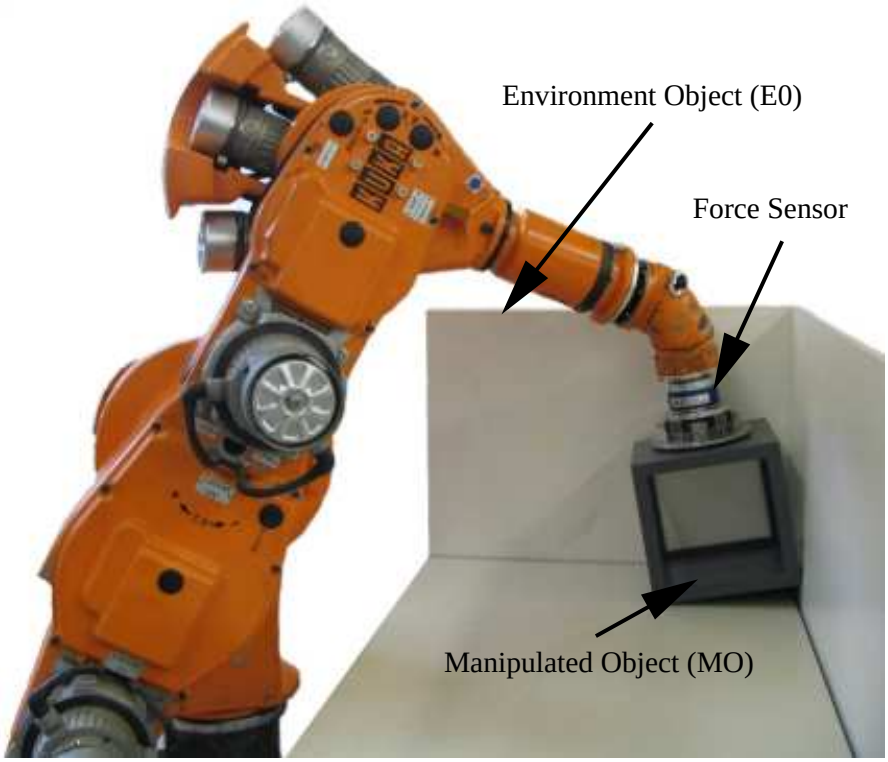


Figure 6.1: Execution of the "cube in corner" autonomous compliant motion experiment. A Kuka 361 serial manipulator, equipped with a force-torque sensor, autonomously assembles a cube (the Manipulated Object MO) into a corner, formed by three perpendicular planes (the Environment Object E0). The unknown continuous parameter variables are the location of the MO with respect to the end effector of the robot, and the location of the EO with respect to a fixed "world frame". These are often referred to as the *geometrical parameters*. The measurements that are used for the estimation of these parameters are the wrenches (force-torque measurements) from the force sensor and the position measurements from the encoders of the robot.

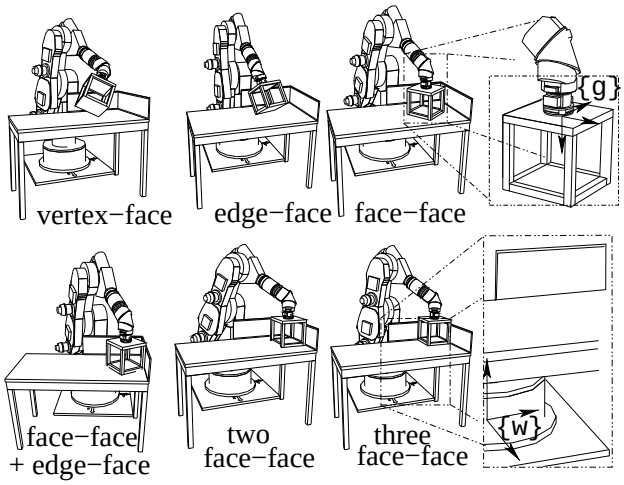


FIGURE 6.2: Execution of the "cube in corner" autonomous compliant motion experiment: different *Contact Formations* (CF).

sensor, should insert a cube (further in this Chapter referred to as Manipulated Object, MO) into a "corner" (Environment Object, EO).

Both the location of the MO (with respect to the end effector of the robot) and the EO (with respect to the environment), and some dimensions of the MO/EO are uncertain. These variables are referred to as the *geometrical parameters*, and they form the *continuous* part of the estimation problem. The location of the end effector with respect to the world frame is assumed to be known, i.e. the uncertainty due to the *kinematic chain* of the robot manipulator (calibration errors, backlash, ...) is negligible compared to the uncertainty on the geometrical parameters. The measurements used for the estimation are wrenches (forces and torques) from a wrist force sensor, and position measurements from the encoders.

The execution of a typical force-controlled task can be segmented into different *discrete* Contact Formations (CF, e.g. a *vertex-plane* contact, an *edge-plane* contact or a *plane-plane* contact): every CF gives rise to a different *measurement model* and control law. In order to be able to estimate the geometrical parameters during the force-controlled execution of CF sequences, it is thus necessary to recognize the current CF. The problem of *CF recognition* includes the simpler problem of *CF transition detection*. Lefebvre, Bruyninckx, and De Schutter (2003) describe both the problem formulation and measurement equations of the presented autonomous compliant motion case.

Compared to previous research (Lefebvre, Bruyninckx, and De Schutter

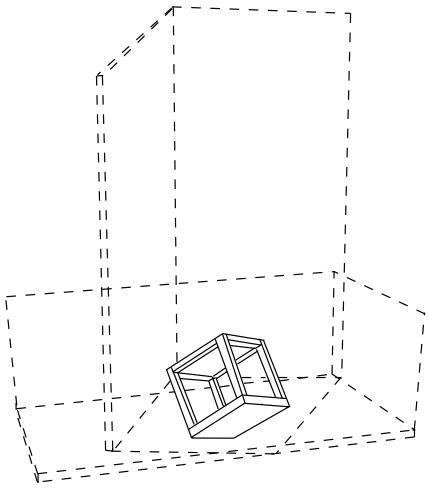


Figure 6.3: Cube in corner assembly under large initial uncertainty of the environment object. Although the informative measurements indicate that a vertex-face contact is established, we don't know which plane of the EO is in contact with the vertex of the cube: two of the many possibilities are shown here.

symb. Ch. 5	meaning	nature	symb. Ch. 6
$\boldsymbol{\theta}$	geometrical parameter vector	continuous	$\boldsymbol{\theta}$
$\boldsymbol{x}$	Contact Formation state	discrete	CF
$\boldsymbol{z}$	measurement (position/wrench)	continuous	$\boldsymbol{z}$

Table 6.1: Meaning of parameters, states, and measurement for the cube-in-corner assembly experiment.

2003), the presented work allows the execution of the task with even larger uncertainty. Indeed, previous research assumes that, whenever a contact state transition is detected by a statistical consistency test on the wrench measurements, the next CF is known. However, if the initial uncertainty on the geometrical parameters is large, there are many possible *next* CFs. This is illustrated for the case of a vertex-face contact in figure 6.3. This work uses the hybrid estimation approach, developed in Section 5.5.2, using an explicit discrete *model for CF transitions* where both *pose and wrench* measurements are used to detect CF state transitions. The meaning of parameters, states, and measurements in this Chapter are summarized in table 6.1.

We argue that Kalman Filter variants cannot cope with the more powerful hybrid models introduced in this work to deal with this increased uncertainty, but we show that Particle Filters can. The uncertainty the robot can cope with

is only limited by the number of particles, the chosen proposal distribution and problem specific optimizations of the Particle Filter algorithm. However, a Particle Filter needs *explicit* measurement equations, and this Chapter shows how to transform available *implicit* measurement equations (kinematic closure and energy dissipation) into explicit ones.

The outline of this Chapter is as follows: Section 6.2 describes previous work in the fields of geometrical parameter estimation, CF transition monitoring and recognition or combinations of both. The hybrid joint posterior density is derived in a formal way in Section 6.3. Section 6.4 describes why Kalman Filter variants, used extensively in previous research, are not able to deal with the developed model. Particle Filters can deal with the extra amount of complexity. Section 6.5 describes the measurement equations for the cube-in-corner assembly task and explains what it takes to use a Particle Filter to represent the hybrid joint posterior. The results of the estimation using a Particle Filter are described in Section 6.6. The Chapter ends with some conclusions and directions for future work. Chapter 7 discusses the software framework in which this application was developed.

6.2 Previous Work

Most authors consider only one of the two problems—the estimation of geometrical parameters assuming the CF is known, or the estimation of CFs without geometrical parameters—, or do not model the interaction between the CF states and the geometrical parameters.

Estimation of geometrical parameters. The identification of the contact location and orientation based on wrench, position and/or twist measurements is often applied for *single point-surface contacts*. The main application area is 2D contour tracking. Only few authors consider contact situations which are *more general than point-surface contacts* (Mimura and Funahashi 1994; Debus, Dupont, and Howe 2002; De Schutter, Bruyninckx, Dutré, De Geeter, Katupitiya, Demey, and Lefebvre 1999). Most authors use *non-recursive deterministic* estimation, i.e., the estimates are based only on the last measured data and no measurement uncertainties are considered.

CF transition monitoring and recognition. CF recognition techniques can be divided into methods based on *learned contact models* and methods based on *analytical contact models*. The learned models can handle uncertainty taken into account at modeling time, i.e. deviations from parameter values for which training data was available. A deterministic, Bayesian, Fuzzy (Skubic and Volz 2000) or Neural Network model (Asada 1993) performs the CF recognition or transition detection. Hidden Markov Models (Hovland and

McCarragher 1998; Hannaford and Lee 1991) are a popular tool for both the detection of transitions and the recognition of the current CF.

Simultaneous CF recognition and geometrical parameter estimation. The CF models are a function of the uncertain geometrical parameters. During task execution, the uncertainty reduces due to the sensor information. This increases the knowledge of the "true state" of the system and environment which improves the force control, the CF transition monitoring and the CF recognition.

Mimura and Funahashi (1994) recognize *vertex-plane*, *edge-plane* and *plane-plane* CFs based on wrench, twist and pose measurements. The different CF models are tested from the least to the most constrained till a model is found which is consistent with the data (i.e., a model for which the geometrical parameters can be determined).

In our lab, for execution of the cube-in-corner task *with small uncertainties*, the executed sequence of CFs is assumed to be error-free, i.e., after a contact transition the CF is the next one in the (off-line calculated) *task plan* (Hirukawa, Papegay, and Matsui 1994; Xiao and Ji 2001). This means that, after an inconsistency detection, two CFs are probable: the same CF as before the inconsistency detection (false alarm) and the next CF in the task plan. This is only valid for small uncertainties. (Lefebvre, Bruyninckx, and De Schutter 2003; De Schutter, Bruyninckx, Dutré, De Geeter, Katupitiya, Demey, and Lefebvre 1999; Lefebvre, Bruyninckx, and De Schutter 2004a) estimate the geometrical parameters with Kalman Filters and detect inconsistency by a SNIS-test (Bar-Shalom and Li 1993). Several Bayesian techniques such as Kalman Filter variants (Kalman 1960; Lefebvre, Bruyninckx, and De Schutter 2004b) and Monte Carlo methods (Doucet, de Freitas, and Gordon 2001; Gadeyne and Bruyninckx 2001) have been used for the estimation of the geometrical parameters, assuming the CF is known. These approaches *only use measurement information* for the detection of a CF transition, and do not develop an explicit model for CF transitions. In (Mihaylova, Lefebvre, Staffetti, Bruyninckx, and De Schutter 2002), a first step towards a multiple model posterior is made.

Debus, Dupont, and Howe (2002) present similar results with *deterministic* multiple model estimation based on pose measurements. The penetration distances of the objects from the different deterministic filters (CFs) are used as measurements in a HMM which has CFs as states. Their HMM approach, although using only deterministic penetration distances (and not the information from the wrenches) as measurements, is similar to the simplified system model of the hybrid joint posterior developed in Section 6.3.

Recently, Slaets et al. (2004) present some results in which a Bayes' Factor approach is applied in combination with a SNIS-test. Their method is more

oriented to model building from a given number of primitives, and the results they obtain are only valid once the posterior has converged to a unimodal Gaussian.

6.3 Bayesian estimation of the hybrid joint density

The autonomous execution—without assuming the CF sequence is known—of the experiment of figure 1.2 requires a *hybrid* (partly continuous, partly discrete) joint posterior Probability Density Function (PDF) representing the belief that, at time step k, the CF[1] is j ($0 \leq j < \#CFs$) and the geometrical parameters $\boldsymbol{\Theta}$ have a certain value $\boldsymbol{\theta}$, given all measurements $\boldsymbol{Z}$ up to k:

$$P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_k = j \mid \boldsymbol{Z}_{1:k} = \boldsymbol{z}_{1:k}\right). \tag{6.1}$$

Figure 6.4 shows an example of such a hybrid joint posterior density, if $\boldsymbol{\Theta}$ would be one-dimensional. Each CF j has "its own" continuous density $P\left(\boldsymbol{\theta} \mid CF_k = j, \boldsymbol{z}_{1:k}\right)$. The difference between $P\left(\boldsymbol{\theta} \mid CF_k = j, \boldsymbol{z}_{1:k}\right)$ and $P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k}\right)$ is only a scale factor independent of $\boldsymbol{\theta}$ (product rule).

In many sensor-based tasks, including the particular case of the above-mentioned assembly, there is no obvious model-based (direct) relation $P(\boldsymbol{z}_k \mid CF_k)$ between the discrete model and the measurements of the sensor. So, one can not use Bayes' rule to find out how much information one has already gathered to support a particular CF:

$$P\left(CF_k \mid \boldsymbol{z}_{1:k}\right) = \frac{P\left(\boldsymbol{z}_k \mid CF_k\right) P\left(CF_k \mid \boldsymbol{z}_{1:k-1}\right)}{P\left(\boldsymbol{z}_k \mid \boldsymbol{z}_{1:k-1}\right)}.$$

However, a measurement model $P(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k = j)$ does exist: given the current CF and the geometrical parameters, we know the probability that a certain measurement of the force-sensor will occur.

Assuming a Markovian system, Bayes' rule allows us to update the joint hybrid posterior eq. (6.1) recursively with each new measurement $\boldsymbol{z}_k$. The following derivation makes this explicit for the hybrid PDF $P(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k})$. Assumptions about the specific nature of the CF recognition and geometrical parameter estimation allow us to simplify some of the general expressions presented in Chapter 2. Applying Bayes' rule to (6.1) yields:

$$P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k}\right) = \frac{P\left(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k = j\right) P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right)}{P\left(\boldsymbol{z}_k \mid \boldsymbol{z}_{1:k-1}\right)}. \tag{6.2}$$

[1] This Chapter uses the symbol CF_k to represent the discrete, one-dimensional state vector $\boldsymbol{x}_k$.

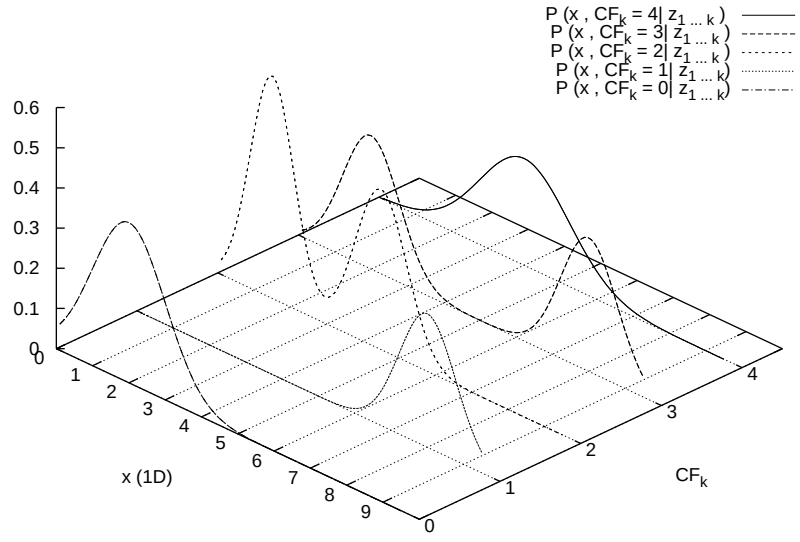


Figure 6.4: Example of a hybrid joint density, with a one-dimensional geometrical parameter $\mathbf{\Theta}$. The CF_k axis represents the Contact Formation at timestep k.

$P(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1})$ is often called the *prediction density*. It describes our belief about the hybrid state $(\boldsymbol{\Theta}, CF_k)$ at time step k, *without* taking into account the information provided by the measurement at that time.

The prediction density $P(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1})$ is related to the posterior at time step $k-1$ by a hybrid *system model* describing transitions from CF i to CF j, *given* the value of the geometrical parameters:

$$\begin{aligned} &P(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}) \\ &\quad = \sum_i P(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i)\, P(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1})\,. \end{aligned} \tag{6.3}$$

The term $P(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i)$ is the probability of a contact transition, given knowledge about the current CF and given that the value of $\boldsymbol{\Theta} = \boldsymbol{\theta}$. Indeed, using the information of the velocity setpoints sent to the motion controller, we can predict CF transitions using the forward kinematics of the robot manipulator and an off-line calculated contact graph (Xiao and Ji 2000; Hirukawa, Papegay, and Matsui 1994). However, the current implementation of the system model only uses the off-line calculated graph to describe the possible transitions between CFs:

$$P\left(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}', CF_{k-1} = i\right) \approx P(CF_k = j \mid CF_{k-1} = i)\,. \tag{6.4}$$

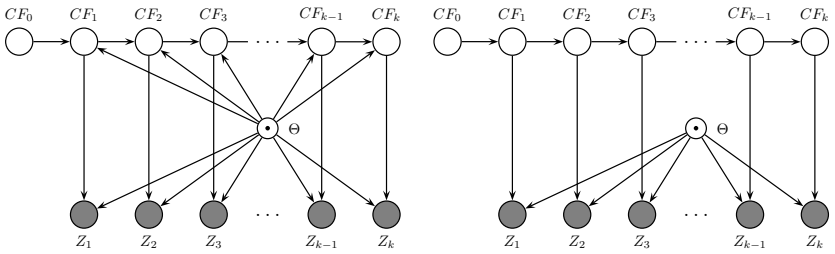
In terms of DBNs, this boils down to the simplification shown in figure 6.5. This simplification has some disadvantages, because it does not correspond well to reality. Indeed, using this model, the probability of staying in a certain CF decreases exponentially with time:[2]

$$P(CF_{\tau+1:\tau+k} = \{j, j, \ldots, j\} \mid CF_\tau = j) = a_{jj}^k, \qquad 0 \leq a_{jj} \leq 1$$

while in reality, the probability of staying in a certain CF depends also on the value of the geometrical parameter vector and the input speed of the end effector. The above mentioned graph only specifies which CF transitions are possible, and does not yield probabilities a_{ij}. However, the experiment described in Section 6.6 shows that the information contained in the measurements is rich enough to compensate this poor system model and the chosen values for the parameters a_{ij} are not critical for obtaining good results.

Note that another way to cope with the issue of CF transition models is to use *model building* from scratch. Slaets et al. (2004) detect several vertex-plane contacts and use hypothesis tests to identify if e.g. two (or more) vertices are in contact with the same plane. This approach does not use the above described CF transition model $P(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i)$.

[2] $\{j, j, \ldots, j\}$ denotes a sequence of k j's.



(a) $P\left(CF_k \mid \boldsymbol{\Theta}, CF_{k-1}\right)$. (b) $P\left(CF_k \mid CF_{k-1}\right)$.

FIGURE 6.5: Simplification of the DBN model of the cube-in-corner assembly task. The model on the left uses a realistic model to describe CF transitions, the model on the right only uses information from an off-line calculate graph that describes possible CF transitions.

However, the results of the hypothesis tests are only valid once the posterior has converged to a unimodal Gaussian.

So, for the autonomous compliant motion case, taking into account assumption (6.4), (6.3) becomes:

$$\begin{aligned} &P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right) \\ &\quad \approx \sum_i \left[P(CF_k = j \mid CF_{k-1} = i) P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1}\right)\right]. \end{aligned} \tag{6.5}$$

$P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1}\right)$ represents the posterior density at time $k-1$.

6.4 Interpretation and discussion

Summarizing, equations (6.2) and (6.5) describe the recursive two–step system and measurement update of the Bayesian joint hybrid posterior for semi-

dynamic[3] systems:

System update:

$$P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right) = \sum_i P(CF_k = j \mid CF_{k-1} = i) P\left(\boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1}\right);$$

Measurement update:

$$P\left(\boldsymbol{\theta}, CF_k \mid \boldsymbol{z}_{1:k}\right) \sim P\left(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k\right) P\left(\boldsymbol{\theta}, CF_k \mid \boldsymbol{z}_{1:k-1}\right). \tag{6.6}$$

The system update describes each prediction density as a weighted sum of the different continuous estimates for each CF at time $k-1$ $P\left(\boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1}\right)$, where the weights $P(CF_k = j \mid CF_{k-1} = i)$ include information from a *system graph*. At each time step, every continuous density in figure 6.4 is replaced by a weighted sum of the N previous estimates (N represents the number of CFs).

Given the joint posterior (6.1), it is straightforward to estimate the current probability distribution over CFs, via marginalisation of the geometrical parameters:

$$P\left(CF_k = j \mid \boldsymbol{z}_{1:k}\right) = \int_x P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k}\right) d\boldsymbol{\theta}. \tag{6.7}$$

The marginal probability $P\left(\boldsymbol{\theta} \mid \boldsymbol{z}_{1:k}\right)$ that the geometrical parameter vector $\boldsymbol{\Theta}$ has a value $\boldsymbol{\theta}$, is expressed as a weighted average over the state estimates of the different CFs:

$$P\left(\boldsymbol{\theta} \mid \boldsymbol{z}_{1:k}\right) = \sum_i P\left(\boldsymbol{\theta}, CF_k = i \mid \boldsymbol{z}_{1:k}\right). \tag{6.8}$$

As with all Bayesian algorithms, several implementations of the above derivation could be used, at least in theory. Section 6.4.1 describes why Kalman Filter variants (Lefebvre, Bruyninckx, and De Schutter 2003; Lefebvre, Bruyninckx, and De Schutter 2004a), used in previous research to estimate the geometrical parameters assuming the CF is known, are not capable of dealing with the more powerful model as described by eq. (6.6). Particle Filters (Gordon, Salmond, and Smith 1993; Doucet, de Freitas, and Gordon 2001) *can* deal with the extra complexity introduced by the hybrid model and this approach is described below in Section 6.4.2.

6.4.1 Kalman Filter variants

The interaction between the estimates from the different CFs—necessary for coping with the extra uncertainty introduced by the fact that CFs are

[3] We use the term semi-dynamic since the CF model is dynamic, but the geometrical parameters are static.

unknown—results in an increasing complexity of the posterior's shape with time. If, at time 0, we start with N (the number of Contact Formations) unimodal Gaussians, at time step k, the posterior PDF of eq. (6.1) consists of N multi-modal Gaussians with N^{k-1} modes, resulting in memory requirements that are quadratic with time. Hence, we have to eliminate most of the modes at each time step to maintain a constant complexity. However, the resulting PDF should be a *consistent* and *informative* approximation (Lefebvre 2003) of the original density. A variety of algorithms have been developed to perform this reduction in the linear Gaussian case (e.g. second order generalized pseudo Bayes (GPB2), Interacting Multiple Model Filtering (IMM), ...). They are sometimes referred to as Assumed Density Filtering (ADF), see e.g. (Bar-Shalom and Fortmann 1988) or (Minka 2004) for an overview.

Application to Cube-in-Corner assembly

Unfortunately, for the assembly problem described in this Chapter, the measurement model—given a certain contact formation— $P(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k)$ is highly non-linear. Previous research focused on Kalman Filter Variants, and has used the Iterated Extended Kalman Filter (IEKF) to solve the problem of estimating the geometrical parameters, given a certain CF (Lefebvre, Bruyninckx, and De Schutter 2003). This limits the execution of the task to execution under relatively small uncertainties due to the linearisation. To allow the execution of the experiment under large uncertainties, the Non-Minimal State Kalman Filter (NMSKF, (Lefebvre, Bruyninckx, and De Schutter 2005a)) was developed. By transforming the geometrical parameters into a higher dimensional space where a linear model is obtained, and solving the estimation problem in that space, this filter avoids the accumulation of errors due to linearisation. However, a different transformation is used for each CF. This means that the multiple model approach approach cannot be applied in the higher dimensional space. Therefore, in the case of highly non-linear measurement models, Kalman Filter variants are not capable of dealing with the increased complexity of the hybrid joint posterior.[4]

6.4.2 Particle Filter approaches

The Particle Filter (Chapter 4) is another possibility to perform inference in this hybrid model. Indeed, sequential Monte Carlo[5] methods use a fixed num-

[4]Note that another inconvenience of the Kalman Filter approach is the fact that Kalman Filter variants cannot deal with the full system model $P\left(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}', CF_{k-1} = i\right)$ from equation (6.4) because of its hybrid nature. Although the presented Chapter does not use this more accurate but complex system model yet, this would be prohibitive for future research.

[5]Because of the online estimation requirements, we only consider Sampling Importance Resampling methods for the parameter estimation, and no MCMC or other computationally

ber of samples to represent an arbitrary complex PDF. Contrary to Kalman Filter variants, Particle Filters can easily track discrete, continuous and hybrid Random Variables. Furthermore, their ability to track multi-modal posteriors and to cope with non-linear measurement equations makes them attractive to use for inference in hybrid non-linear models. However, they have also several disadvantages with respect to Kalman Filters. Their computational complexity is significantly higher than that of Kalman Filter derivatives, especially in this case where the state vector is 12-dimensional (or even higher if some of the dimensions of the objects are unknown).

Complexity of the Particle Filter algorithm

Let N be the number of CFs, and M the total number of samples to represent the hybrid joint density. Every continuous estimate belonging to a certain CF $P(\boldsymbol{\theta}, CF_k = i \mid \boldsymbol{z}_{1:k})$ will be represented by a number of samples M_i proportional to the marginal densities $P(CF_k = i \mid \boldsymbol{z}_{1:k})$:

$$M_i = P(CF_k = i \mid \boldsymbol{z}_{1:k}) \times M, \qquad i = 1, \ldots, N. \tag{6.9}$$

The prediction step calculates M samples from the proposal density ($O(M)$). The measurement update multiplies each of the M samples with a weight obtained from the measurement equation. The resampling step draws M_i independent samples from each density $P(\boldsymbol{\theta}, CF_k = i \mid \boldsymbol{z}_{1:k}))$ in $O(M_i)$ through ordered uniform sampling (Section 3.3).

Consequently, the total time needed for updating the posterior every time step will be $O(M)$ for a basic Particle Filter algorithm. Memory requirements are of the same order.

The resulting algorithm, in the case of a Bootstrap Filter with dynamic resampling (Section 4.4.1), is described in algorithm 6.1. Note that artificial dynamics are introduced on $\boldsymbol{\theta}$ in order to avoid degeneracy (Section 5.4.1).

Section 6.5.1 describes the measurement equations for this assembly problem. However, these equations are implicit, and Particle Filters need *explicit* equations for the weight update step. The derivation of explicit equations is dealt with in Section 6.5.2. Finally, Section 6.5.3 presents a discussion on the modeled uncertainties in the particular case of this assembly task.

6.5 Implementation

6.5.1 Measurement equations

Lefebvre et al. (2003) describe the specific measurement equations of this autonomous compliant motion case in full detail. Therefore, this Section focuses

more intensive methods.

Algorithm 6.1 Particle Filter algorithm for the hybrid posterior with dynamic resampling and the system model as proposal density.

Sample N samples from the a priori density $P(\boldsymbol{\theta}_0, CF_0)$
for $i = 1$ to N **do**
  $\tilde{w}_0^i = \frac{1}{N}$
end for
for $k = 0$ to T **do**
  for $i = 1$ to N **do**
    Sample CF_k^i from $P(CF_k \mid CF_{k-1})$
    Sample $\boldsymbol{\theta}_k^i$ from $P(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1})$ (artificial dynamics)
    Assign the particle a weight according to the measurement update: $w_k^i = P(\boldsymbol{z}_k \mid \boldsymbol{\theta}_k^i, CF_k^i)\tilde{w}_{k-1}^i$
  end for
  Normalize the weights:
  for $i = 1$ to N **do**
    $\tilde{w}_k^i = \frac{w_k^i}{\sum_{i=1}^N w_k^i}$
  end for
  Calculate the effective sample size $ESS = \frac{1}{\sum_{i=1}^N (\tilde{w}_k^i)^2}$
  if $ESS < threshold$ **then**
    Resample
  end if
end for

on the *filter–specific* issues of these measurement equations. The definition of the used frames is shown in figure 6.6. The parameter vector $\boldsymbol{\theta}$ is twelve-

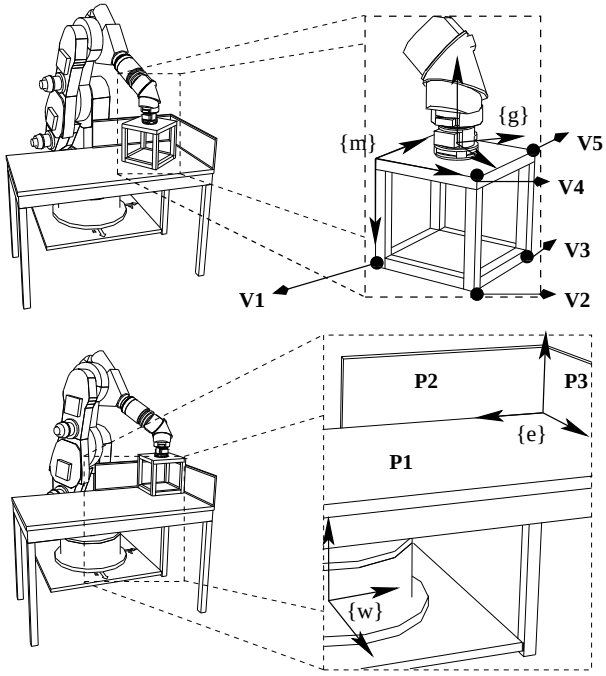


FIGURE 6.6: Frames and vertex/plane numbering for the Cube in Corner experiment: $\{e\}, \{w\}, \{g\}, \{m\}$ denote Environment, World, Gripper, and MO Frame respectively.

dimensional and contains (i) the position and orientation of the MO (frame $\{m\}$) with respect to the gripper ($\{g\}$) and (ii) the position and orientation of the EO ($\{e\}$) with respect to the world frame ($\{w\}$). The measurement vector contains wrenches (6D), twists (6D) and the position/orientation of the gripper $\{g\}$ with respect to the world frame $\{w\}$.

Each CF can be composed of a number of elementary CFs, e.g. a *plane-plane* CF can be decomposed into three *vertex-plane* contacts. This allows automatic generation of the measurement equations for the different CF models (Lefebvre, Bruyninckx, and De Schutter 2003). Both wrench measurements from the force sensor and pose measurements from the encoders give rise to *non-linear implicit* measurement equations $h(\boldsymbol{\theta}, \boldsymbol{z}) = \mathbf{0}$ (eqs. (6.10) and (6.12)).

The wrench measurement equations are based on the approximation that there is no friction during the experiment, and thus the dissipated energy (for

each *vertex-plane* contact) vanishes. For each CF, only a limited subset of the 6D space of wrenches is possible: e.g. figure 6.7 illustrates that a single *plane-plane* contact without friction cannot produce a torque around an axis perpendicular to the planes in contact, nor a force parallel to that plane. The vector space of all possible contact wrenches for a certain CF is called

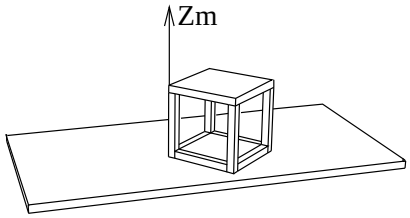


FIGURE 6.7: A single *plane-plane* contact without friction cannot produce a torque around an axis Zm perpendicular to the planes in contact, nor a force parallel to that plane.

the *wrench space*. Each of those contact wrenches can be written as a linear combination of the vectors in the *wrench base* $\boldsymbol{G}$. Similarly, a *twist base* $\boldsymbol{J}$ describes a minimal spanning set for all possible twists in a given CF. Twists and wrenches are reciprocal: In a direction of frictionless motion, one cannot measure a wrench—the dissipated energy vanishes—, as expressed by the wrench measurement equation:

$$\boldsymbol{E} = \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{w} = \boldsymbol{0}. \tag{6.10}$$

The number of columns of $\boldsymbol{J}$ (and thus the number of rows of the $\boldsymbol{0}$ vector) varies along the given CFs. $\boldsymbol{J}(\boldsymbol{\theta})$ is a non-linear function of the parameter vector $\boldsymbol{\theta}$.

The pose measurement equations express the closure of the kinematic chain robot–MO–EO; e.g. for a *vertex-plane* contact this means the vertex lies in the plane and results in (one equation per *vertex-plane* contact):

$$\boldsymbol{n}^T \, {}_e\boldsymbol{p}^{e,c} = d, \tag{6.11}$$

where ${}_e\boldsymbol{p}^{e,c}$ is the location of the vertex c expressed in $\{e\}$; and the location of the plane in $\{e\}$ is given by its normal vector $\boldsymbol{n}$ and d is the (perpendicular) distance from the origin of $\{e\}$ to the plane.[6]

Since we only know the position of the vertex in $\{m\}$ we have to transform

[6] Written down in a scalar form, eq. (6.11) is equivalent to $ax + by + cz - d = 0$, where $\boldsymbol{n} = [a\ b\ c]^T$ is normal to the plane and the coordinates of the vertex in the appropriate frame are $[x\ y\ z]^T$.

${}_m\boldsymbol{p}^{m,c}$ to $\{e\}$. This results in the *closure equations*:

$$\begin{bmatrix}\boldsymbol{n}\\0\end{bmatrix}^T \; {}^w_e\boldsymbol{T}(\boldsymbol{\theta})\; {}^g_w\boldsymbol{T}(\boldsymbol{z})\; {}^m_g\boldsymbol{T}(\boldsymbol{\theta}) \begin{bmatrix} {}_m\boldsymbol{p}^{m,c}\\1\end{bmatrix} = d, \tag{6.12}$$

where $\boldsymbol{T}$ denotes a homogeneous non-linear transformation matrix between the different frames.

6.5.2 Explicit equations for Particle Filters

Particle Filters require explicit measurement equations: at each timestep they calculate the probability of the new measurement $P(\boldsymbol{z}_k \mid \boldsymbol{\theta})$ for each particle. Kalman Filters do not use this probability value. Therefore, it is not possible to use the same measurement equations as in previous research. The following paragraphs explain how the existing implicit measurement equations (constraints) are transformed into explicit equations.

Wrench measurement equations Assuming additive Gaussian uncertainty on the wrench measurements $\boldsymbol{w}$, and taking into account that $\boldsymbol{J}(\boldsymbol{\theta})$ in (6.10) is known for a given particle state value $\boldsymbol{\theta}$, $\boldsymbol{E}$ is a linear transformation of the wrench vector $\boldsymbol{w}$, for a given particle state value $\boldsymbol{\theta}$. This means that the uncertainty on $\boldsymbol{E}$ also will be additive Gaussian:

$$P(\boldsymbol{E} \mid \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{\Sigma}_w \boldsymbol{J}(\boldsymbol{\theta})), \tag{6.13}$$

where $\boldsymbol{\Sigma}_w$ denotes the original covariance matrix on $\boldsymbol{w}$ and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multi-variate Gaussian with mean value $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

The "energy measurement vector" $\boldsymbol{E}$ represents the dissipated energy in each of the directions allowing motion freedom. This means the number of rows of $\boldsymbol{E}$ depends on the number of elementary contacts in a given CF and $\boldsymbol{E}$ is a different random variable for each of the CFs. So, it is impossible to compare different CF hypotheses with this approach! However, since wrenches and twists are reciprocal, the combined twist-wrench measurement equation

$$\begin{bmatrix}\boldsymbol{J}^T & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{G}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{w}\\ \boldsymbol{t}\end{bmatrix} = \boldsymbol{0}, \tag{6.14}$$

where $\boldsymbol{t}$ denotes the twist vector, does represent the "instantaneously dissipated energy" in *all* possible directions (of dimension six) and can be used to compare different CF hypotheses. Equation (6.13) then becomes

$$P(\boldsymbol{E} \mid \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{J}(\boldsymbol{\theta})^T & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{G}(\boldsymbol{\theta})^T\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\Sigma}_w & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Sigma}_t\end{bmatrix} \begin{bmatrix}\boldsymbol{J}(\boldsymbol{\theta})^T & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{G}(\boldsymbol{\theta})^T\end{bmatrix}\right), \tag{6.15}$$

where $\boldsymbol{\Sigma}_t$ denotes the original covariance matrix on $\boldsymbol{t}$.

Pose measurement equations The closure equations (6.12) express the fact that, per *vertex-plane* contact, the perpendicular distance from the vertex to the plane should vanish. These equations are implicit, suggesting the application of a technique similar to the one used in the above paragraph, with the perpendicular distance to the plane d as a new, "derived" measurement variable. However, unlike in the case of the twist-wrench measurement eq. (6.14), for a given value of $\boldsymbol{\theta}$, the transformation between the pose measurement (denoted as $\boldsymbol{z}$ in eq. (6.12)) and d is still non-linear. This means that additive Gaussian uncertainty on the pose measurements will no longer be additive Gaussian uncertainty on the transformed distance measurement.

The original choice of adding additive Gaussian uncertainty to the pose measurements in previous research was due to the fact that additive Gaussian uncertainty is a necessary assumption for the Kalman Filter. This assumption is justified by the Central Limit Theorem that states that any sum of *many* independent identically distributed random variables is *approximately* normally distributed. The uncertainty is the sum of (i) position errors on the six joint positions of the robot, (ii) the compliance of the MO, EO and force sensor, (iii) measurement noise and discretisation errors in the encoders, and (iv) errors in the kinematics. Since this assumption also holds for the perpendicular distance d from the origin of $\{e\}$ to the plane, adding Gaussian uncertainty to the distance vector instead of the pose measurement vector is justifiable.

A second problem with this transformation is the difference in dimensions of the transformed measurement vector between different CFs. For each vertex plane contact, an equation of the form (6.12) can be written. The solution is found by treating all CFs as combinations of vertex plane contacts: e.g. for a single *plane-plane* (SPP) contact the following holds during the cube-in-corner assembly:

$$\begin{aligned} &P(CF = \text{SPP}) \\ &= P(\text{V1 in P1}, \text{V2 in P1}, \text{V3 in P1}, \text{V4 } \textbf{not} \text{ in P2}, \text{V5 } \textbf{not} \text{ in P2}, \text{V5 } \textbf{not} \text{ in P3}), \end{aligned} \tag{6.16}$$

where Vi denotes Vertex i and Pj denotes Plane j as shown in figure 6.6. For each individual vertex plane contact, we assume

$$P(d \mid \boldsymbol{\theta}, \text{Vi in Pj}) = \mathcal{N}(d_j, \sigma_{ij}). \tag{6.17}$$

Note that σ_{ij} could depend on $\boldsymbol{\theta}$. d_j denotes the perpendicular distance from the origin of $\{e\}$ to the Plane j. If Vi is not in Pj, we assume d is also normally distributed, but with a another mean value d'_j and another covariance matrix. So, if the vertex is not in contact with the plane, we have

$$P(d \mid \boldsymbol{\theta}, \text{Vi not in Pj}) = \mathcal{N}(d'_j, \sigma'_{ij}). \tag{6.18}$$

Note that d'_j is not equal to d_j, and will depend on $\boldsymbol{\theta}$.

For each CF, we can construct a measurement vector from r transformed pose measurements (r is the minimal number of single vertex-plane contacts in the most constrained CF, and $r = 6$ in the case of the cube in corner application). The resulting probability is the product of the "individual" Gaussian probabilities, e.g. for a single *plane-plane* contact:

$$\begin{aligned} &P(\boldsymbol{z}' \mid \boldsymbol{\theta}, CF_k = SPP) \\ &= P(d_1 \mid \boldsymbol{\theta}, CF_k = SPP) P(d_2 \mid \boldsymbol{\theta}, CF_k = SPP) \dots P(d_5 \mid \boldsymbol{\theta}, CF_k = SPP) \\ &= P(d_1 \mid \boldsymbol{\theta}, V1 \text{ in } P1) P(d_2 \mid \boldsymbol{\theta}, V2 \text{ in } P1) \dots P(d_5 \mid \boldsymbol{\theta}, V5 \text{ not in } P3). \end{aligned} \tag{6.19}$$

6.5.3 Uncertainty on the models

The relative pose uncertainty originating from the (calibrated) robot is much smaller than the uncertainty on the wrench measurements. This is due to the nature of the wrench sensor and the model errors (such as friction) in the wrench measurement model. Therefore, the stochastical estimator mainly uses the information of pose measurements for its estimation of the geometrical parameters, i.e. the uncertainty on the pose measurements determines the accuracy of the estimates of the geometrical parameters.

Previous research (Lefebvre, Bruyninckx, and De Schutter 2003) used wrench measurements mainly to detect CF transitions. It only took into account the possibility of a CF transition when the SNIS test failed, due to an inconsistency of wrench-twist measurements with the model. This research also uses the pose models to detect CFs.

6.6 Experimental results

This Section describes the experimental results of the estimation of geometrical parameters and CFs during a cube in corner assembly task with a KUKA-IR 361 robot arm with a Particle Filter. Following the principle of *reproducible research* (See Chapter 7), all source code and the measurement data for reproducing these results is available via the BFL homepage (Gadeyne 2001b). The established CFs during the experiment are the *vertex-plane* CF (CF0), the *edge-plane* CF (CF1), the *plane-plane* CF (CF2), the *plane-plane* + *edge-plane* CF (CF3), the 2 *plane-plane* CF (CF4) and the 3 *plane-plane* CF (CF5).

At time 0, we know for sure that the cube is in free space ($P(CF_0 = CF0) = 1$). We suppose $P(\boldsymbol{\Theta} = \boldsymbol{\theta} \mid CF_0 = CF0)$ is a multi-variate Gaussian, with the initial values as described in (Lefebvre, Bruyninckx, and De Schutter

2003) to allow comparison of the results: The parameters of the MO and EO are uncorrelated. The chosen means are

$$\mu_{EO_\theta} = [-0.07\ -0.06\ -3.2]^T, \mu_{EO_{pos}} = [0.4\ -0.65\ -0.75]^T,$$
$$\mu_{MO_\theta} = [0.01\ 0.01\ -0.78]^T, \mu_{MO_{pos}} = [0.122\ 0.127\ -0.0155]^T$$

(rotations are represented as Euler-ZYX angles, expressed in radians, positions in m).[7] The true values of the parameters were

$$\mu_{EO_\theta} = [0\ 0\ -3.1415]^T, \mu_{EO_{pos}} = [0.457\ -0.608\ -0.680]^T,$$
$$\mu_{MO_\theta} = [0\ 0\ -0.785]^T, \mu_{MO_{pos}} = [0.125\ 0.125\ -0.015]^T.$$

The chosen covariances for rotational uncertainty are (in degrees)

$$\Sigma_{EO_\theta} = \begin{bmatrix} (10^\circ)^2 & 0 & 0 \\ 0 & (10^\circ)^2 & 0 \\ 0 & 0 & (10^\circ)^2 \end{bmatrix}, \quad \Sigma_{MO_\theta} = \begin{bmatrix} (1^\circ)^2 & 0 & 0 \\ 0 & (1^\circ)^2 & 0 \\ 0 & 0 & (1^\circ)^2 \end{bmatrix}.$$

For position uncertainty, we chose

$$\Sigma_{EO_{pos}} = \begin{bmatrix} (100mm)^2 & 0 & 0 \\ 0 & (100mm)^2 & 0 \\ 0 & 0 & (100mm)^2 \end{bmatrix},$$

and

$$\Sigma_{MO_{pos}} = \begin{bmatrix} (1mm)^2 & 0 & 0 \\ 0 & (1mm)^2 & 0 \\ 0 & 0 & (1mm)^2 \end{bmatrix}.$$

Section 6.6.1 describes the results of the CF recognition using only the twist and wrench measurement, section 6.6.2 describes the results of both CF recognition and parameter estimation when using both measurement models.

6.6.1 CF Estimation using only twist and wrench data

Using only the information from the twist and (noisy) wrench data, the Particle Filter cannot estimate the geometrical parameters but should be able to recognize the different CFs. Figure 6.8 shows the results of a C++ program with 50000 samples, using the Bayesian Filtering Library (see Chapter 7) on a 1.13 GHz/128Mbytes RAM pc, running Linux.[8] Artificial dynamics were

[7] Refer to Section 6.5.1 and figure 6.6 for the exact definition of the geometrical parameter vector.

[8] For this problem, the same results were obtained with as few as a 100 particles. One inconvenience of Particle Filters is the fact that, to the best of our knowledge, there's currently no algorithm that can predict how many samples must be applied in order to achieve meaningful results using a Particle Filter. The only solution is trial and error. Without any optimizations, ± 500 samples can be used to process the measurements at a rate of $10Hz$.

introduced on the parameters to avoid degeneracy problems (Section 5.4.1). The algorithm is the one described in algorithm 6.1.

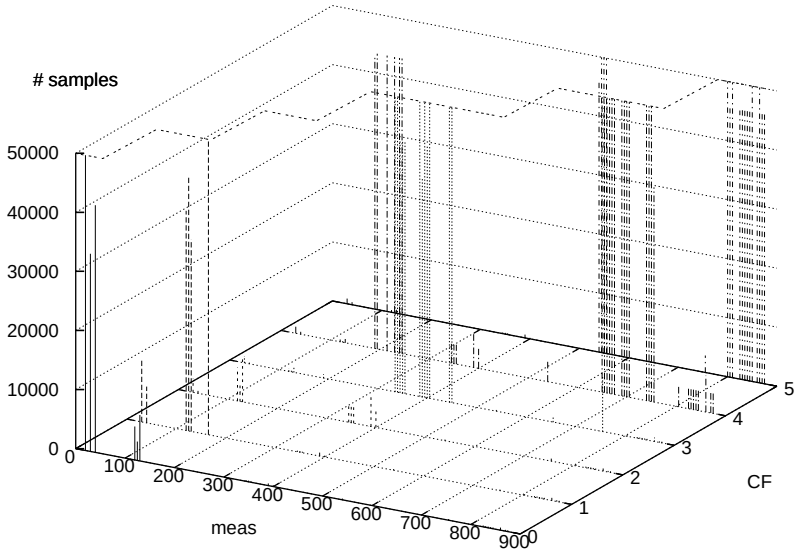


FIGURE 6.8: True CF evolution (The dashed–line curve) and sample based estimate using a Bootstrap Filter with 50000 samples. The "meas" axis denotes the number of the processed measurements, the "# samples" axis shows the number of samples in a particular CF. So a cross section parallel to the plane formed by the "# samples" and the "CF" axis, and crossing the "meas" axis at time k, represents the marginal posterior density $P(CF_k|z_{1...k})$. Similar results are obtained using a Particle Filter with as few as 100 samples.

The Figure illustrates that the Bootstrap Filter (Section 4.4.1) estimates the CF reasonably well when contacts are well established (otherwise no model is available), except in the *plane-plane* CF (CF2, between measurements $180 - 280$) and a part of the 2 *plane-plane* CF (between measurements $506 - 560$). Both "errors" can be explained given the values of the twist measurements during those periods (figure 6.9). The Particle Filter identifies the *plane-plane* CF as a 2 *plane-plane* contact. This is caused by the fact that the angular velocities in the extra degrees of freedom of the *plane-plane* CF (with respect to the 2 *plane-plane* contact) are virtually zero: The *plane-plane* CF allows a rotation around the axis perpendicular to the plane (denoted as the "Rot z" in figure 6.9) but the robot doesn't use this degree of freedom. So, based on these measurements, one *cannot* distinguish between

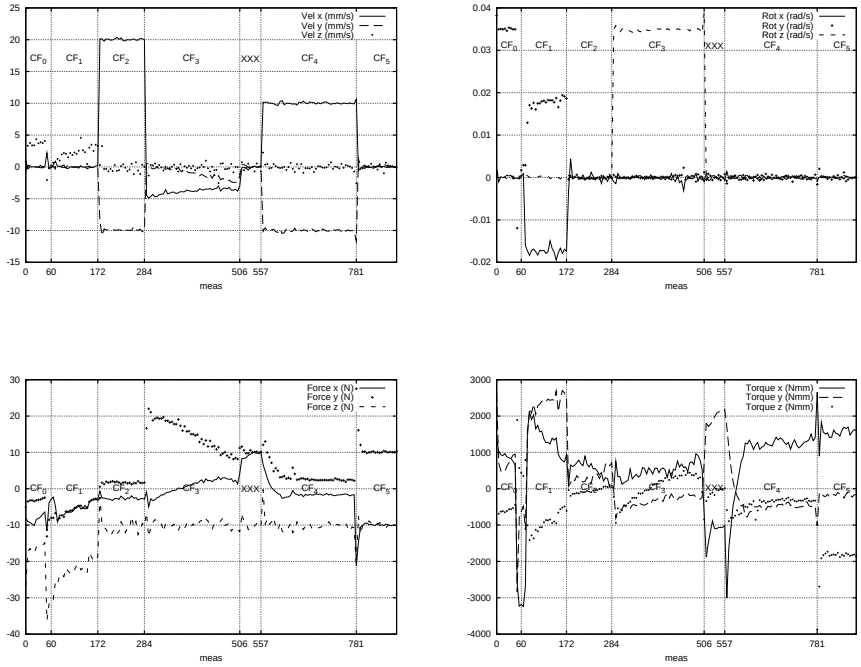


Figure 6.9: Translational and angular velocity (above) and wrench (below) measurements during cube-in-corner experiment. The XXX denotes the part of CF3 where the robot does not move.

the *plane-plane* (CF2, contact measurements $172-284$) and the 2 *plane-plane* contact (CF4, measurement $557-781$) given only the twist measurements. Occam's razor (See Section 5.4.3) then explains why the Particle Filter assigns more probability to the 2 *plane-plane* CF by looking at the difference in the covariance on $\boldsymbol{E}$ between the 2 CFs. Due to the much higher uncertainty, wrench measurements are not significantly present in the filter's estimates. As an example, suppose the current CF is the *free space* contact. This CF has a twist base $\boldsymbol{J}$ of six columns and an empty wrench base. Eq. (6.15) is then equivalent to (6.13). The resulting covariance matrix on $\boldsymbol{E}$ will be large, since the covariance matrix on $\boldsymbol{w}$ is also large (compared to the covariance on the twist measurements). On the other hand, consider the 3 *plane-plane CF*: $\boldsymbol{J}$ is now empty, whereas the wrench basis $\boldsymbol{G}$ consists of six columns. So, (6.15) simplifies to $P(\boldsymbol{E} \mid \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{G}(\boldsymbol{\theta})^T \boldsymbol{\Sigma}_t \boldsymbol{G}(\boldsymbol{\theta}))$. The covariance matrix on $\boldsymbol{E}$ is smaller, since the covariance matrix on $\boldsymbol{t}$ is small compared to the covariance on the wrench measurements (and $\boldsymbol{J}$ and $\boldsymbol{G}$ are of the same order of magnitude). If velocities in certain directions are low, both CFs explain these measurements. Occam's razor will prefer the "simpler" model with the smaller covariance.

The error during the 2 *plane-plane* CF (identified by the Particle Filter as a 3 *plane-plane* CF) is also explained using Occam's razor given the fact that the robot does not move at all during this time period (see figure 6.9 during measurements $506-557$).

When increasing the initial uncertainty a lot, we notice the same tendency to favor CFs with less degrees of freedom. This would not happen if (i) the uncertainty on the wrench measurements was smaller, e.g. if a friction model would be available, and/or (ii) the motion degrees of freedom were better excited (for the first situation)!

The effect of changing the parameters of the system model is minor, suggesting that the information contained in the measurements is rich enough to compensate the approximate system model of Section 6.3. The presented results are obtained using the following CF transition matrix:

$$\begin{bmatrix} 0.95 & 0.03 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.02 & 0.95 & 0.02 & 0.0033 & 0.0033 & 0.0033 \\ 0.0033 & 0.02 & 0.95 & 0.02 & 0.0033 & 0.0033 \\ 0.0033 & 0.0033 & 0.02 & 0.95 & 0.02 & 0.0033 \\ 0.0033 & 0.0033 & 0.0033 & 0.02 & 0.95 & 0.02 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.03 & 0.95 \end{bmatrix}.$$

6.6.2 Simultaneous Geometrical Parameter estimation and CF recognition using Position and Wrench/Twist information

CF recognition

Implementing the pose measurement model described in Section 6.5.2 requires the choice of parameters μ and σ in both equations (6.17) and (6.18). Although a very rough approximation of reality,[9] we chose

$$d'_j = d_j, \tag{6.20}$$

where d_j is the (perpendicular) distance of plane Pj to the origin of the environment frame $\{e\}$ and

$$\sigma_{ij}^2 = (2mm)^2 \qquad (\sigma')_{ij}^2 = (300mm)^2. \tag{6.21}$$

These approximations appear sufficient (again, using Occam's razor principle) to distinguish between the two situations (in contact or not) and to estimate (a part of) the geometrical parameter vector if Vi is in contact with Pj. Figure 6.10 shows the result of such a simulation with 400 samples. The complementary information in both measurement models allows to estimate the CF far better than using only the Twist-Wrench model. Indeed, in the two cases described above, the extra information stemming from the position measurements allows to choose the right CF. Note also, that compared to figure 6.8, the marginal posterior density at time-step k, $P(CF_k|z_{1\ldots k})$, are far less "spread out" along the CF-axis. This is logical due to the relative quality of the position measurements with respect to the wrench/twist information. The extra computational cost by also using the position measurements is neglegible.

Geometrical parameter estimation

Using information of both measurement models, the Particle Filter also provides estimates for the geometrical parameters. Figure 6.11 shows the results for the geometrical parameter estimates of the EO, obtained by marginalizing the hybrid posterior from the Particle Filter. The algorithm yields similar results as provided by an Iterated Extended Kalman Filter (IEKF), where the segmentation and CF detection still was done manually! The only noticeable

[9] In reality, the mean distance d_j and the covariance σ_{ij}^2 will be dependent on i and j (the indices of vertex and plane in contact, respectively). So, using only position information and the approximative values in those equations, a transition in CF is made sometimes too soon, sometimes too late. So, only the *combination* of the force/twist and position allows to perform the CF recognition "seamlessly".

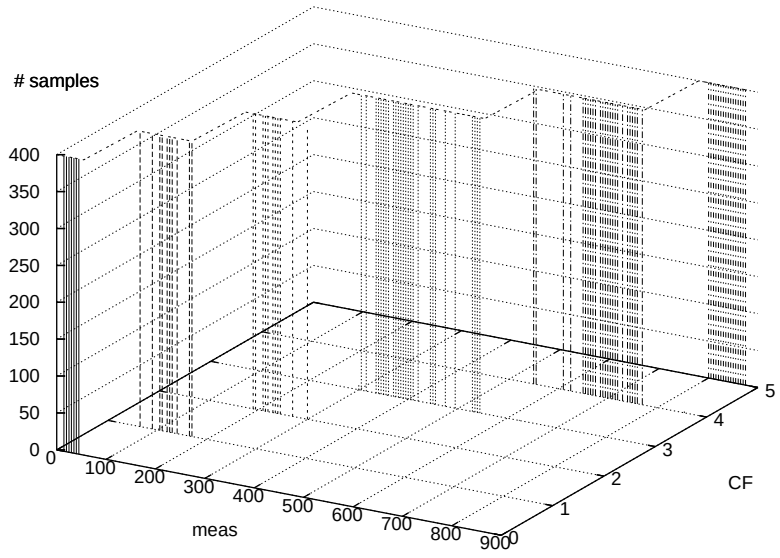


FIGURE 6.10: CF recognition with information from both measurement models. Compared to figure 6.8, the results are better, because of richer sensor information.

differences occur when for parts of the parameter vector which are unobserved. These are due to the artificial dynamics on the parameters, resulting in a random walk behaviour as long as no information is available. The "bad" estimation result for the x component is probably also due to this random walk behaviour, in combination with the fact that this component only becomes observable at the very end of the experiments. This means there's not enough information available yet to compensate for the uncertainty introduced by the (longish) random walk.

In this work, the segmentation is fully automatic. Note that not all parameters are observable in all CFs: e.g. a single *vertex-plane* contact does not yield any information about x^e, y^e or θ_z^e. This explains why most estimates only converge to the *true* values after a large number of measurements. For this particular case, running the algorithm with 5000 samples (which are necessary to provide a reasonably good estimate of the geometrical parameters) takes about 1500 seconds on a 1.1 GHz laptop (to process the 900 measurements which correspond to 90 seconds of real experiment time). This means that an optimized version of the code could easily process measurements at a rate of 1 Hz.[10] Furthermore, since algorithm 6.1 used a basic Particle Filter with a simple proposal density, using better proposal densities would lead to significant improvements in accuracy (or obtaining the same accuracy faster with less particles). This is a topic of future research.

6.7 Summary and Conclusions

This Chapter presents simultaneous model-based estimation of geometrical parameters and Contact Formation recognition in environments with a significant amount of uncertainty. Previous research (Lefebvre, Bruyninckx, and De Schutter 2003) only used wrench measurements to detect CF transitions. This work develops a hybrid (partly discrete, partly continuous) approach, using an explicit discrete transition model for CFs where both pose and wrench measurements are used to detect CFs. This allows to deal with larger uncertainties in both CF and continuous geometrical parameter estimation. The Chapter argues that Kalman Filter variants, used in previous research for the estimation of geometrical parameters given a certain model, are not able to cope with this increased level of complexity. Indeed, the Iterated Extended Kalman Filter can only handle small uncertainties due to the severe nonlinearities in the models, and the Non Minimal State Kalman Filter cannot deal with model selection. Particle Filter implementations successfully estimate the hybrid posterior density.

[10] Recent optimisation efforts for BFL, mainly the work from Wim Meeussen, have lead to a speed increase of about factor 10, meaning that measurements could be easily processed at a rate of 10 Hz on a Pentium 4 CPU.

The CF recognition problem is similar to the data association problem in SLAM-like applications, but there is a major difference: in mobile robot localisation problems, the mobile robot controller is not directly interested in the value of the "data association" variable, while for force-controlled assembly that value is of utmost importance in order to select a safe control law that fits with the current contact constraints.

This Chapter advocates the use of a hybrid posterior instead of the popular Maximum Likelihood approach in mobile robotics for this particular problem, due to the nature of the compliant motion problem and the importance of a correct contact recognition.

Explicit measurement models for the Particle Filter, based on energetic and closure constraints are developed, leading to good experimental results for both the CF recognition and the estimates of the twelve-dimensional parameter vector. The increased complexity due to the use of the hybrid posterior is acceptable for this particular case.

Ongoing research focuses on:

- The use of better models for the description of the transition behavior between CFs.
- A better choice of proposal distribution of the Particle Filter (local linearisation of measurement model), taking into account the last measurement such as the Extended Kalman Particle Filter or the Unscented Particle Filter.
- *Active sensing* is also envisioned, i.e. imposing motion trajectories and CF transitions to speed up the estimation of the geometrical parameters to a desired accuracy.

The first two items should allow to further reduce the sensitivity of the algorithm to the choice of some parameters under *larger* uncertainties without increasing the number of samples.

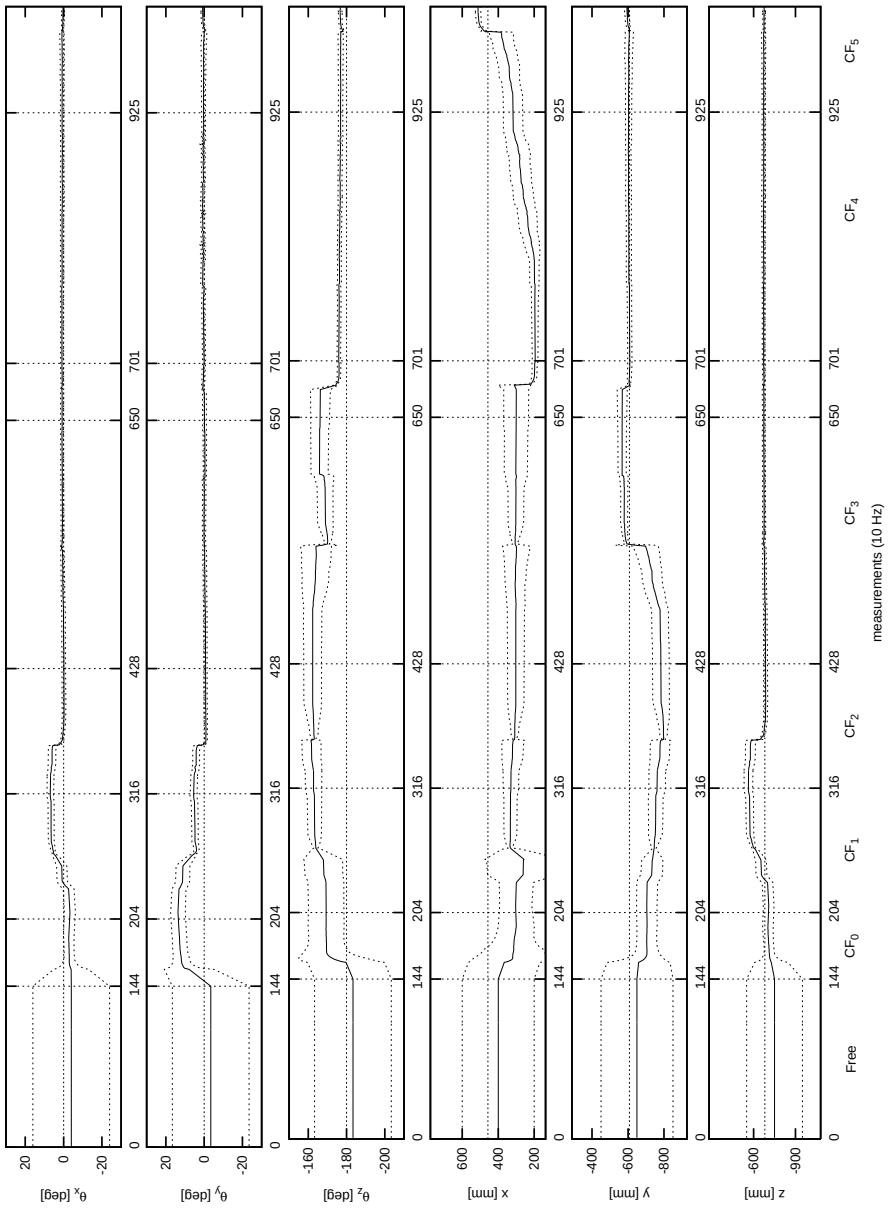


Figure 6.11: Particle Filter estimates of the geometrical parameters of the Environment Object. The straight horizontal dotted line denotes the true state. The full line shows the posterior mean of the Particle Filter and the curved dashed lines denote the 2σ boundary. 50000 samples were used for this figure, 5000 samples are sufficient to assure convergence.

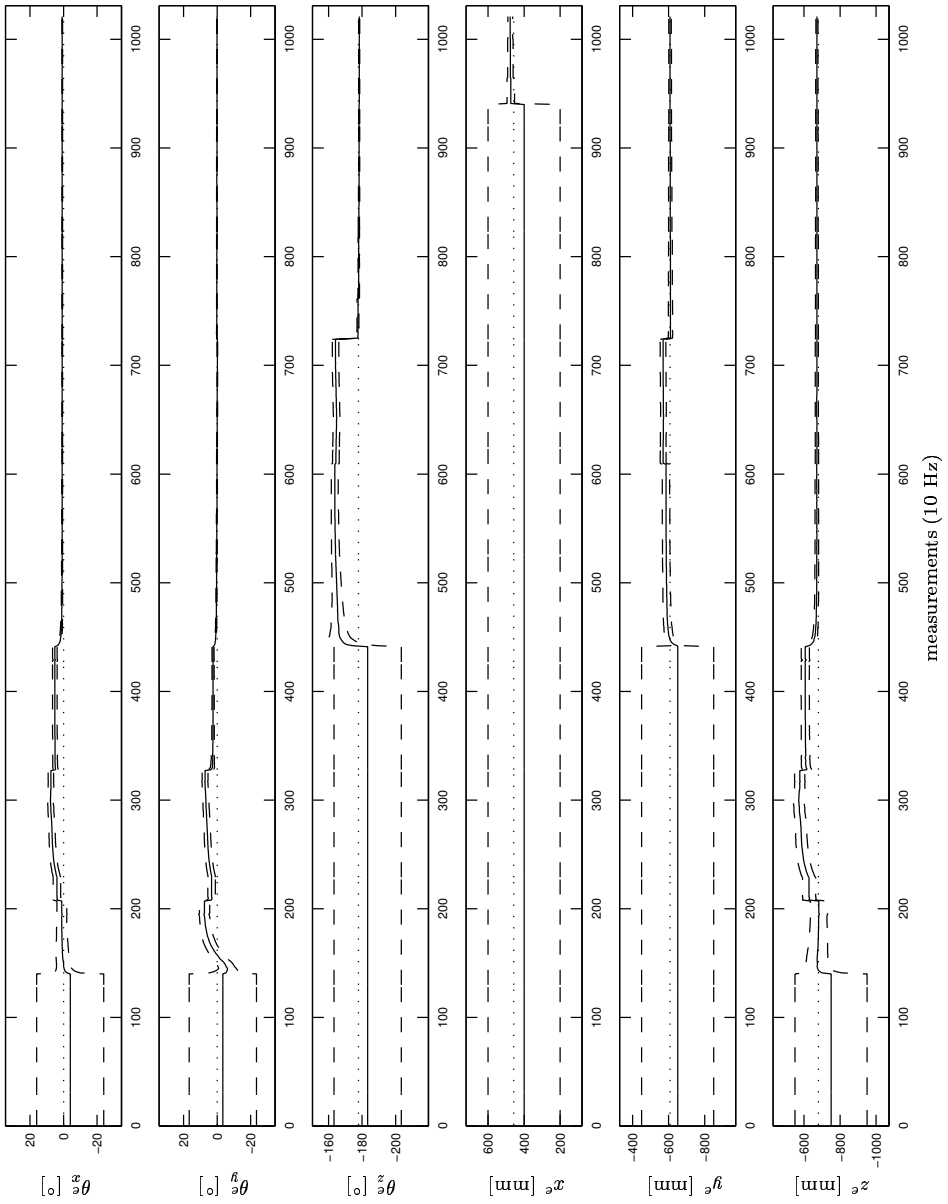


FIGURE 6.12: Iterated Extended Kalman Filter estimates of the geometrical parameters of the Environment Object, with *manual segmentation of the CFs*. The straight horizontal dotted line denotes the true state. The full line shows the posterior mean of the IEKF and the curved dashed lines denote the 2σ boundary.

Chapter 7

BFL: an Open Software Framework for recursive Bayesian Filtering

7.1 Introduction

As the previous Chapters demonstrated, a lot of research is done in the area of Bayesian state and parameter estimation. Consequently, a lot of software implementing this approach is available: libraries for Kalman Filtering, grid based filtering, Monte Carlo filtering, Bayesian Networks, Hidden Markov Model algorithms, ... Unfortunately, most of them have been designed with one particular application in mind, making the library hard to reuse in other context. Section 7.2 describes the requirements that form the reason for existence of the *Bayesian Filtering Library* (BFL). Several other libraries implementing Bayesian Filtering algorithms are available, Section 7.3 examines how closely they fulfil the stated requirements. The basic design of BFL is illustrated in Section 7.4. Some applications using BFL are sketched in Section 7.5.

7.2 Requirements

BFL (Gadeyne 2001b) has been designed with the following requirements in mind:

Bayesian The goal of the framework is to provide a fully Bayesian software framework, i.e. all Bayesian methods described in Chapter 5 should fit in the

library design. This implies that the library should not impose restrictions (i) on the nature of the Random Variables—both discrete, continuous and hybrid states/parameters— (ii) nor on the representation of the posterior PDF (analytical, sample based, ...).

The common Bayesian background of these methods allows to implement different Bayesian algorithms with a maximum of code reuse. Moreover, the performance of these algorithms can be compared with a minimum effort.

Open One of the problems comparing different models and algorithms in robotics, is the lack of standardized data sets. The *Robotics Data Set Repository* (Radish) (Howard and Roy 2003) is a first attempt to spread common data sets, currently strongly biased to mobile robot localization. A common software framework for all Bayesian inference algorithms, with a maximum of code reuse, is ideal to study differences between different algorithms or different implementations of a particular algorithm. The Open Source Initiative (Open Source Initiative) offers an enormous potential in this scope. Furthermore open source software is a key factor implementing the idea of *reproducible research* (Buckheit and Donoho 1995).[1] Indeed, published experimental results often only show a snapshot of the "*whole truth*", that is achieved for certain parameter values, initial conditions, ... The state of the art can only benefit from the availability of the source code, in order to reproduce the obtained results and to gain better understanding by altering certain chosen values of the experiments.

Independent At present, there is no *standard* numerical nor stochastical library for C++ available, such as the Standard Template Library (STL) (Musser and Saini 1996). Therefore, there's a wide range of libraries available providing the necessary functionality. An estimation library is usually only a part of the whole software infrastructure in order to perform a certain task. To avoid ending up with multiple libraries for one project, the ideal Bayesian estimation library should be as decoupled as possible from one particular numerical/stochastical library.

A second meaning of independent is the independence of a particular application. The filtering framework should be independent of any particular application (such as Autonomous Compliant Motion, mobile robotics, econometrics, ...). This means both its interface and implementation should be decoupled from particular sensors, assumptions, algorithms, ... that are specific to a certain application.

[1] My personal opinion is that *reproducible* research is usually a synonym for *better* research.

Furthermore, the library is to be integrated in our robot control software written in C++ and therefore, C++ is chosen as programming language.

7.3 Other libraries

This Section discusses several software initiatives that fulfil one or more of the requirements described in the previous Section.

7.3.1 Bayes++

Bayes++ (Stevens 2003) is an open source C++ library implementing several Bayesian Filters. It has the same scope of BFL, in casu online inference in Dynamic Bayesian Networks. Bayes++ distinguishes between models and algorithms, but it doesn't implement a PDF interface. Furthermore, it only supports continuous state and parameter values. The library focuses more on optimization than on the use of generic interfaces that facilitate interoperability. Bayes++ uses the Boost libraries (Boost) as Matrix and stochastical libraries and the Boost `bjam` utilities to compile on both Linux and Microsoft OSes.

7.3.2 Scene

Scene (Davison and Kita 2001) is a C++ library for online inference in DBNs. Although the software seems to be alive, the latest official release is from June 2001, and other code is not yet available.[2] The 2001 release is biased towards mobile robotics applications (SLAM) where cameras are used as sensors, and towards Kalman Filter variants from an algorithmic point of view. Scene is tied to one particular numerical library (Horatio) and is also limited to online inference for fully continuous systems.

7.3.3 Bayesian Networks software libraries

Murphy (2004b) maintains a list of software for inference in (static) BNs. Based on the Bayes Net Toolbox (BNT) for Matlab (Murphy 2001), Intel announced an open source C++ version of the BNT, OpenPNL (Intel) in late 2003. OpenPNL was recently (December 2004) opened for external developers and currently includes the code for inference in static Bayesian Networks, some support for inference in Dynamic Bayesian Networks (i.e. the same functionality as BFL), and for ML learning. Unfortunately, documentation is somewhat lagging behind. However, given its foundations are based on Bayesian Networks, it seems like a promising alternative for the future.

[2] Checked February 2005.

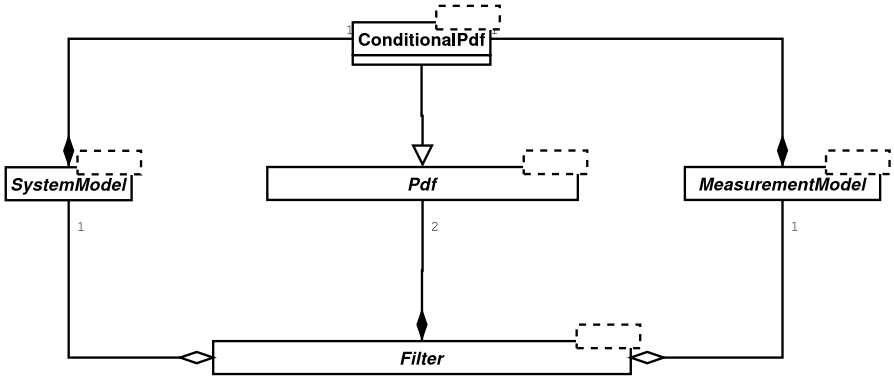


Figure 7.1: Basic design of the BFL, represented by a UML diagram.

7.3.4 CES

Although CES (C (Thrun 1998b) or C++ (Thrun 2000) for Embedded Systems) is a prototype of a programming *language* and not a software library, its design largely[3] fulfils the requirement of providing a Bayesian framework. Indeed, CES includes language support for computing with uncertain information and support for (off-line) learning. The C++ version claims to support continuous and discrete variables by a template system. Goncalves et al. (2003) describe the use of CES for embedded mobile robotic systems. However, to the best of my knowledge, CES' design and source code is not open and it needs a separate (unavailable) compiler.

7.4 Overview of BFL

This Section presents the major design decisions, as well as the functionalities, of BFL.

7.4.1 Class interface design

BFLs basic classes *interfaces* are shown in the UML diagram of figure 7.1. Each Bayesian filter implements the `Filter` interface. The prior and posterior PDF are of implementations of the `Pdf<T>` interface, where `<T>` is a template type. Parameters and states are not represented by different random variable types, as it is up to the programmer to choose the most appropriate filter. So a PDF $P(\boldsymbol{x}, \boldsymbol{\theta})$ with a continuous state vector and discrete parameter is represented by a `Pdf<Hybrid>` interface class in BFL, where `Hybrid` is a class

[3] According to (Thrun 1998b), CES represents PDFs always as a set of samples.

implementing a partly continuous, partly discrete variable. Note that `Pdf` is an abstract interface class, containing no information about the *representation* of the PDF, which can be analytic, sample based, ...

Conditional PDFs are represented by the `ConditionalPdf<T1,T2>` interface class. It represents likelihood models $P(A|B)$, where the RV A is of type `T1` and B is of type `T2`. E.g. the `LinearAnalyticConditionalGaussian` class is an implementation of `ConditionalPdf<ColumnVector,ColumnVector>` interface class, representing likelihood models of the type

$$P(\boldsymbol{y}|\boldsymbol{x}), \qquad \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{\Sigma}\right), \tag{7.1}$$

where both $\boldsymbol{x}$ and $\boldsymbol{y}$ are continuous vectors (represented by the `ColumnVector` type) and $\boldsymbol{A}$ represents a matrix expressing the linear relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Conditional PDFs are the basis of both model classes, `SystemModel<T>` and `MeasurementModel<T1,T2>`. The former only has one template parameter, since it expresses a relationship between $(\boldsymbol{\theta}, \boldsymbol{x}_k)$ and $(\boldsymbol{\theta}, \boldsymbol{x}_{k-1})$, which are obviously of the same type. The latter has two template parameters, one expressing the type of the measurements, the other one the type of the combined parameter/state RV.

Each filter implementation implements the `Filter<T1,T2>` interface, where `T1` and `T2` represent the types of the combined state/parameter vector and the measurements, respectively. A filter contains a Prior and Posterior PDF, and uses system and measurement models to update its information about the unknown Random Variables.

This class interface design makes sure that any type of PDF and RV can be represented by BFL. Therefor, all online fully Bayesian filtering algorithms discussed in Chapter 5 can be implemented in its framework. Further information about BFL' s design, source code and full API documentation can be found at (Gadeyne 2001a).

7.4.2 Abstraction layers

BFL uses software *abstraction layers* (often denoted as *wrappers*) to avoid dependence on one particular numerical or stochastical library, so to allow programmers to use BFL with the library of their choice. Currently both the Newmat matrix library (Davies 1994) (free for all uses) and the matrix library of LTI-Lib—a C^{++} library used in image processing and computer vision—(Alvarado, Doerfler, and Canzler) (licenced under the LGPL (Stallman)), can be used as underlying matrix libraries.

The Scythe Statistical library (Martin, Quinn, and Pemstein) (licenced under the GPL (Stallman)) is currently wrapped and used as stochastical

library.[4]

7.4.3 Open Source software

BFL is released under an open source licence, in casu the GNU Lesser General Public Licence (LGPL) (Stallman). The main reason behind this is the fact I strongly believe in the concept of reproducible research. All source code used to produce the results in this thesis is downloadable via the WWW.

Furthermore, as noted by Eric S. Raymond: "Given many eyes, all bugs are shallow". The open source nature of BFL also forces me and others to properly write and document the source code.

7.5 Applications

This Section *briefly* discusses some applications of BFL programmed during this thesis. Other applications and movies of the applications below are found on its website. BFL is used at universities (and other places) throughout the world, including the universities of Louvain-La-Neuve, Liège, Hong Kong doing research in image processing, and the universities of Malaga and Genova for mobile robot localisation. Note that these experiments are not challenging estimation problems as such, but only serve to illustrate the power and flexibility of BFL.

7.5.1 Mobile robot localisation

Figure 7.2 is a visualization of a BFL Particle Filter used for tracking a mobile robot (with simple motion and measurement equations) in simulation.[5] The robot is localizing itself with respect to a wall parallel to the X-axis, using a Bootstrap Filter (see Section 4.4.1). The robot can only measure the distance *perpendicular* to the wall, and obtains no information about its position along the X-axis. The starting position (i.e. the Prior PDF) is a Gaussian with a mean value of $[0m\ 0m\ 45°]^T$ and a small covariance matrix.[6] The robot moves with a constant linear and angular velocity. For this simple problem, switching between a Kalman Filter and a Particle Filter requires only one line of code to be altered.

[4] Actually, only the Random Number Generation part of the library is used in BFL.

[5] Note that the purpose of this experiment is rather to prove BFL's flexibility when testing different filtering algorithms than to provide a solution to the mobile robot localization problem!

[6] I.e. this is a case of position tracking, and not the more difficult problem of global localisation.

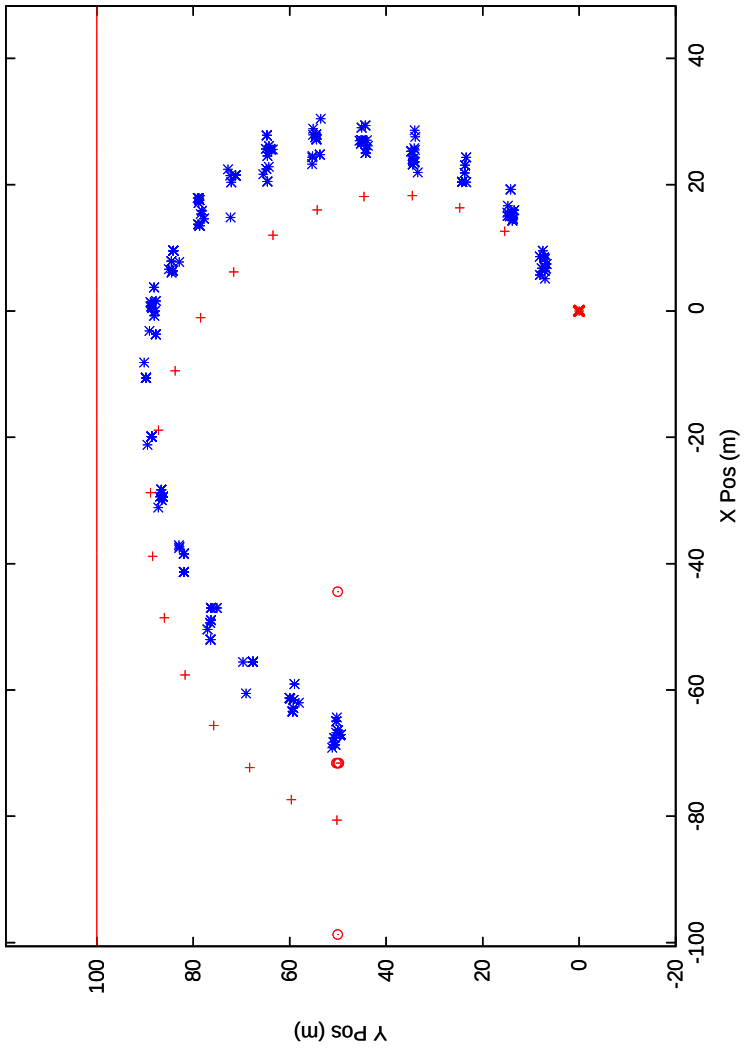


FIGURE 7.2: Mobile robot Localisation using a Simple Particle Filter with 25 samples. The "+" signs denote the simulated states used to generate the measurements. Measurements do not contain information about the position along the X-axis and about the rotation of the robot. The horizontal line at $y = 100$ denotes the location of the wall. Particles are represented by stars ("$*$"). The posterior estimate of a Kalman Filter solving the same problem is shown by circles ("$\circ$"). Five circles are shown, one for the position of the mean value, and four circles indicating the 2σ boundaries. It is obvious that the robot only gets information about its position along the Y-axis, therefore three circles are overlapping.

More background on the system and measurement equations of this applications is found in (De Schutter, De Geeter, Lefebvre, and Bruyninckx 1999).

7.5.2 Simultaneous Contact Formation and Geometrical Parameter recognition

BFL was also used to estimate the hybrid Posterior density for the Simultaneous Contact Formation and Geometrical Parameter recognition, described in full detail in Chapter 6. Note that unlike the previous mobile robotics localisation problem, the Particle Filter requires other measurement equations than the Kalman Filter for this problem.

A movie of the particle cloud during the estimation process is available via BFL's homepage.

Furthermore, BFL has also been used in the model building applications described in (Slaets, Lefebvre, Bruyninckx, and De Schutter 2004).

7.5.3 Tracking a plane with an XY-platform

BFL has been integrated in the *Orocos* robot control software (Soetens and Bruyninckx 2005). The proof of concept is the estimation of the orientation and offset parameters of a plane underneath an XY-platform. To this end, a laserscanner distance sensor is mounted on the cart of the XY-platform, measuring the distance to a plane positioned under the XY-platform. The setup is shown in figure 7.3. The measurements at every timestep are: (i)

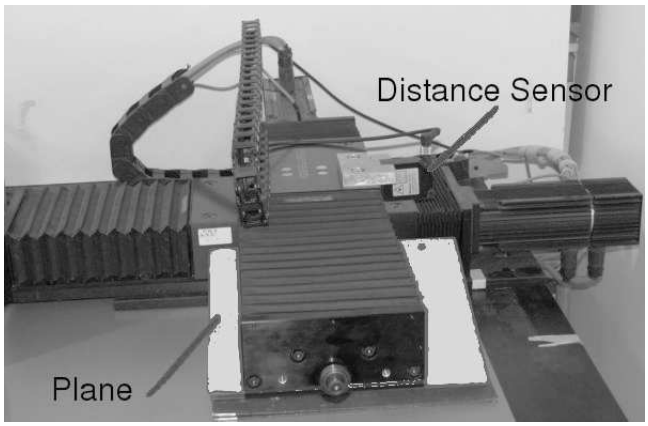


Figure 7.3: Setup for the plane tracking Application. A laser distance sensor is mounted on the cart of an XY-platform. The location of the plane is estimated from distance measurements from the laser scanner setup.

the (x, y) position of the cart, and (ii) the distance d to the plane. The position of the cart is controlled by an Orocos control kernel in realtime using the RTAI/LXRT realtime Operating System (RTOS) (Mantegazza) at 1000 Hz. A second control kernel, running in a non-realtime thread at a lower frequency contains the BFL code. Both Kalman Filter and Particle Filter variants have been used for this parameter estimation case. Note that this particular problem formulation can be modeled with a linear measurement equation

$$z = ax + by + c \quad \text{or} \quad z = [x \ y \ 1] \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{7.2}$$

where a, b, c represent the unknown parameters describing the location of the plane, and x and y are assumed to be known exactly since their position is measured *synchronously* with the distance measurement in the realtime part of the control loop. This allows the use of the optimal importance density for the Particle Filter as described in Section 4.3.1.

In the case the plane is translated and rotated by a human at arbitrary moments in time, the unknown parameters become states varying in time. To achieve a faster tracking performance, some Gaussian *system noise* is inserted:

$$\begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix} = \begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}). \tag{7.3}$$

The covariance of the system noise is a trade-off between the accuracy of the estimates and the tracking performance.

The Orocos property system allows an easy specification of Prior distributions and parameters such as the number of samples to be used for the Particle Filter. This results in a transparent and fast means of comparing different algorithms models and parameters. Estimation results can be transferred and visualized remotely via CORBA (Open Management Group) over Ethernet.

BFL and Orocos are also used together in Programming by Demonstration applications (Rutgeerts 2005) in the Autonomous Compliant Motion group at PMA. More information and movies of these applications are at its homepage (Gadeyne 2001b).

7.6 Conclusions

BFL is an open source C^{++} library for recursive, fully Bayesian inference in Dynamic Bayesian Networks, released under the GNU Lesser General Public Licence (LGPL).

Its main features are (i) a fully templated design and (ii) its notion of a PDF. These allow to cover all fully Bayesian estimation algorithms for recursive state and parameter estimation described in Chapter 5. Its open source code base allows to easily compare research results from different labs, i.e. it adheres the ideas of reproducible research. The use of a numerical and stochastical abstraction layer allows independence of one particular software library and avoids ending up with multiple numerical libraries when integrating several pieces of software.

The current version, 0.4.2,[7] includes support for several Kalman Filter algorithms, such as the Extended and Iterated Extended Kalman Filter, the Square Root Filter (Kaminsky, Bryson, and Schmidt 1971), the Non-Minimal State Kalman Filter (Lefebvre, Gadeyne, Bruyninckx, and De Schutter 2003) and various Particle Filter algorithms such as the standard Bootstrap Filter (using the system model as proposal density), the ASIR filter (Pitt and Shephard 1999), the Extended Kalman Particle Filter (Julier and Uhlmann 2004), and the optimal Proposal Particle Filter.[8] All fully Bayesian algorithms, regardless of their PDF representation and their applicability to a subclass of random variables (discrete, continuous or hybrid) should fit in its framework.

More information, API documentation, mailinglist and applications can be found at BFL's homepage (Gadeyne 2001b).

[7] Released in January 2005.

[8] For models that allow the formulation of the optimal proposal.

Chapter 8

Conclusions

8.1 Introduction: situation of the work

The original aim of this thesis was to find out if the Autonomous Compliant Motion (ACM) research performed at this department could benefit from the relatively new field of sequential Monte Carlo methods. To be able to perform ACM tasks in unstructured environments, e.g. for one-off tasks, online (i.e. recursive and real-time) estimation of unknown parameters/states must be performed. This work focusses on—but is not limited to—the estimation of continuous *geometrical parameters*, such as unknown positions and orientations, and discrete *Contact Formation* states, in the 6D space of rigid body positions and orientations.

Sequential Monte Carlo methods are recursive Bayesian estimation algorithms that represent the posterior probability density function (PDF) as a set of independently and identically distributed (i.i.d.) random weighted samples. These samples can be used to approximate characteristics of the posterior. Although Particle Filter algorithms still lack theoretical convergence results, they already proved to be useful in many applications with highly nonlinear measurement models, as found in mobile robotics, time series estimation, classification problems, etc.

Previous research developed implicit Bayesian models relating the position and wrench measurements from the robot to the unknown geometrical parameters and Contact Formations (CFs). The Non-Minimal State Kalman Filter (NMSKF) (Lefebvre 2003) provides an accurate estimate of posterior PDF over the unknown geometrical parameters, if those parameters are observable and the posterior has converged to a unimodal Gaussian density. However, for large initial uncertainties, the true CF cannot easily be determined (e.g. by the SNIS test used in previous research), and during the initial

measurements, the posterior is typically multi-modal. Moreover, when performing active sensing under large uncertainties, the possibility to detect if the posterior is multi-modal can aid to determine the best possible sensing strategy.

The following Section describes the major achievements of this thesis with respect to these problems. Section 8.3 goes into detail about the limitations of this work and provides some guidelines for future research.

8.2 Contributions

To start with a philosophical note: Perhaps the meta level achievement of this PhD is its loyalty to Bayesian probability theory in all facets: literally from a theoretic point of view, but perhaps more importantly practically, making all assumptions made to obtain the results explicit, both on the algorithmic and software level, hereby promoting the idea of reproducible research.

8.2.1 Classification of Bayesian inference based on nature of the random variables

Chapter 5 presents a cross disciplinary overview of algorithms for estimation of the joint posterior density $P(\boldsymbol{x}, \boldsymbol{\theta})$. Contrary to, and complementary with, existing reviews of inference in Dynamic Bayesian Networks (DBNs) that classify algorithms based on their representation of the posterior, the classification in this thesis is based on the nature of the unknown random variables (discrete, continuous or hybrid, i.e. partly discrete, partly continuous). Important "degenerate" cases are pure state estimation or pure parameter estimation. The focus is on recursive Bayesian estimation algorithms, but links with off-line *training* algorithms to build models and semi-deterministic approximations are made.

The Chapter provides a detailed comparison of algorithms for inference originating from different research areas, presented in a single framework and explains the assumptions that should hold in order to successfully apply these algorithms. Table 5.1 constitutes a "reference card", that helps to choose what algorithms apply once the need for a particular model is identified. This allows experts in a certain application domain to define their needs and choose algorithms without investing too much time finding their way in the "chaos" of available Bayesian models and algorithms.

The Chapter clarifies the relationship between the choice of models and the complexity of implementations and shows how *implicit* measurement models can be considered as a problem of *continuous data association*, an extension of the *discrete* data association problem dealt with in several research areas.

The classification based on the nature of the unknown random variables also exposes that only a limited subset of available models is (frequently) used. The models presented in Chapter 6 and the next Section are an example of situations where new hybrid Bayesian models are used to describe a certain task more accurately. Analytic filters cannot perform inference in such models, but Particle Filters can. Until now, the new research area of Particle Filters has mainly focused on performing inference in *existing* Bayesian models. However, this thesis shows that their flexibility opens new possibilities in the field of Bayesian modeling.

The classification also forms the basis of the Bayesian Filtering Library summarized in Section 8.2.4.

8.2.2 Autonomous Compliant Motion

This thesis develops an explicit (hybrid) Bayesian model for modeling Contact Formation transitions. Previously, transitions were detected by a SNIS test (this research group) or by the use of Hidden Markov Models. The SNIS test does not *model* CF transitions, it only detects inconsistencies. HMMs are a very rough approximation of the CF transition model. The hybrid transition model developed in this thesis emerges naturally from the fully Bayesian recursive derivation of the joint hybrid posterior density $P(\boldsymbol{\theta}, CF_k)$. It *predicts* Contact Formation transitions *based* on the current CF CF_k, and the values of the geometrical parameters $\boldsymbol{\theta}$:

$$P(CF_{k+1}|CF_k, \boldsymbol{\theta}). \tag{8.1}$$

The joint hybrid posterior description allows to deal with larger uncertainties on the geometrical parameters, since it allows to represent ambiguity in the discrete CF variable and to link the measurements to updates of the CF probabilities. The hybrid transition model allows to predict CF transitions more accurately than before.

Sequential Monte Carlo methods can handle both the representation of the hybrid posterior and the hybrid transition model, at the cost of extra computational complexity. Chapter 6 shows how to transform the available *implicit* measurement equations from previous research (kinematic closure and energy dissipation equations) into explicit ones that can be used by the Particle Filter.

The experiments show that Particle Filters succesfully estimate unknown CF states and geometrical parameter simultaneously, where previously applied techniques based on the Kalman Filter fail due to the hybrid nature of the problem.

8.2.3 New hybrid Bayesian models

The hybrid transition model developed for the ACM experiment can also be applied to hybrid state estimation (i.e. replacing the continuous parameter vector by a continuous state vector). This is an extension to Jump Markov Models (often denoted as Jump Markov Systems (JMS)), a model frequently used in literature for hybrid state estimation problems. Section 5.3.3 describes how this hybrid model could be applied to combined state estimation and fault diagnosis.

Figure 8.1 shows a Dynamic Bayesian Network representation of these two new models, relating them to their Bayesian counterpart models that do not allow cross dependency between the value of unknown discrete state and that of the unknown continous parameter or state vector. The main reason that these models have not been used before is probably that analytical filters cannot deal with the hybrid transition models and that these models originate from an earlier period than the revival of Particle Filter methods.

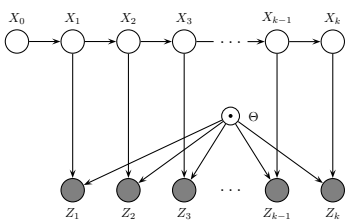
8.2.4 Software

The structure revealed by classification of Bayesian algorithms for recursive inference sketched in Chapter 5 was fundamental in the design of a flexible and modular software framework: the library should not impose restrictions (i) on the nature of the Random Variables—both discrete, continuous and hybrid states/parameters— (ii) nor on the representation of the posterior PDF (analytical, sample based, ...).

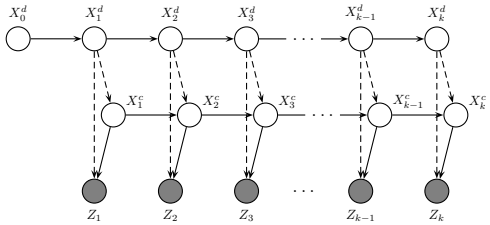
BFL fulfils these requirements. Its fully templated design allows to cover all distinct cases presented in table 5.1 and its abstract notion of a probability density function (PDF) allows to separate the Bayesian concept of a PDF from its representation. This results in a maximal code reuse when comparing different filtering algorithms on a particular DBN topology. In this sense, BFL is also to be considered as a *framework* rather than a *library*. The number and size of its interfaces are small, but any Bayesian algorithm that fits in table 5.1 can be added to BFL.

Furthermore, BFL uses software abstraction layers to achieve independence of any particular numerical and stochastical library. This eases the integration in existing software projects that are already tied to particular libraries. The library also clearly separates the algorithms/models from the application, and is hence independent of any particular application. BFL is an open source project, thereby adhering to the ideas of reproducible research. Its open source nature should also help to improve the state of the art.

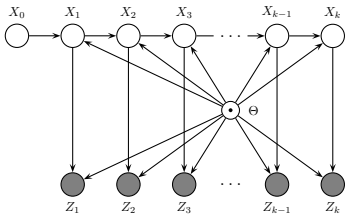
The latter fact is also demonstrated by the integration of BFL in the Orocos software (Soetens and Bruyninckx 2005) independently developed at the department, thereby setting a first step towards the "fully integrated robot



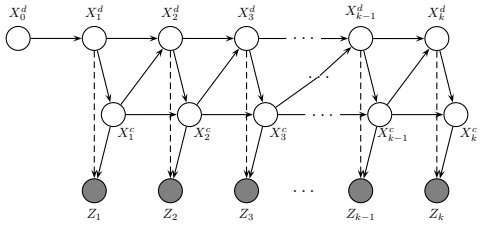
(a) Hidden Markov Model with unknown parameters. In this model, the evolution of the discrete states cannot depend on the value of the unknown parameters.



(b) Jump Markov System. The evolution of the *discrete* hidden states does not depend on the value of the *continuous* hidden states.



(c) Hidden Markov Model with unknown continuous parameters and cross dependency between parameters and states.



(d) Extension of Jump Markov System. This model allows cross dependency between the discrete and continuous part of the state vector.

FIGURE 8.1: Explicit hybrid Bayesian models, allowing cross dependency between the discrete model state and the continuous parameter or state vector versus their counterparts without cross dependency: the Hidden Markov Model with unknown parameters and the Jump Markov System.

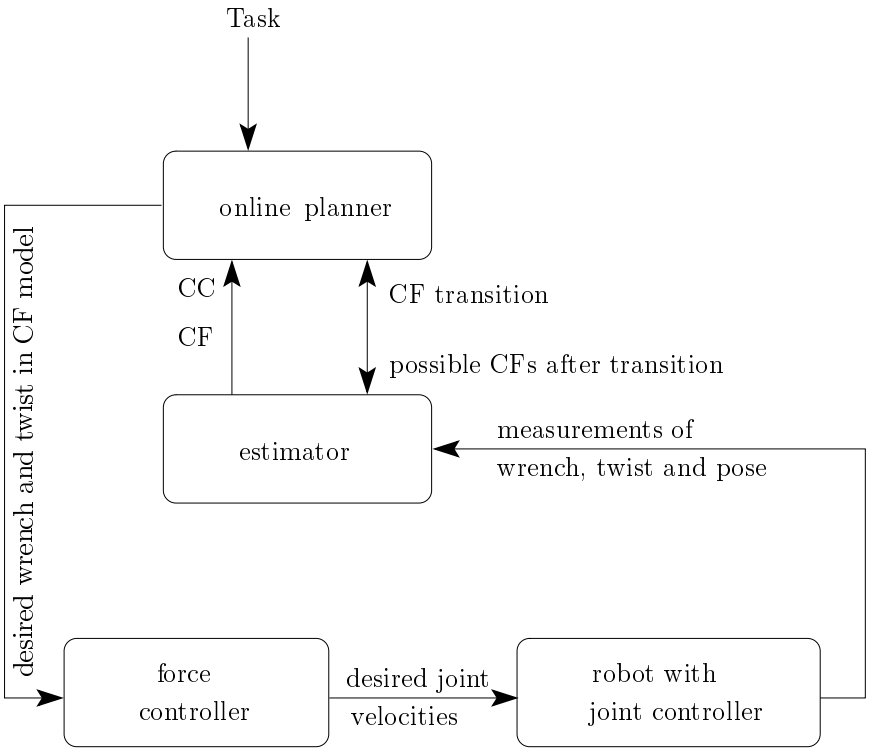


Figure 8.2: The "control" scheme of the ACM system at PMA, consisting of three interacting components: control, estimation and online planning.

controller" shown in figure 8.2. Combined with the Orocos property system that allows an easy specification of prior densities and model parameters, this should result in software applications that can be used by non-Bayesian but application domain experts to test the performance of different algorithms and models in a minimum of time.

8.3 Limitations and future research

Despite its merits, this work constitutes only a small step towards a fully autonomous compliant motion system.

Larger uncertainties. Although, for the uncertainties presented in this thesis, the unoptimized software nearly ran at 0.5Hz using 5000 samples on

a 1.1 GHz pc, I expect that significant performance improvements can be achieved. This will be especially useful and necessary when dealing with larger uncertainties at a higher measurement rate, more complex geometries, higher dimensions, more Contact Formations, *and* considering the fact that estimation is only one part of the ACM application. This leaves the following issues open for further work:

- Implementation of the hybrid transition model $P(CF_{k+1}|CF_k, \boldsymbol{\theta})$. This involves incorporating the velocity setpoints from the controller and the forward kinematics of the robot into the system model.
- Search for better proposal densities. If larger uncertainties are considered, it will be of the utmost importance that the information contained in the measurement is taken into account in the proposal density to avoid explosion of the number of particles necessary to achieve convergence. The Extended Kalman Particle Filter and the Unscented Kalman Filter seem good alternatives here.

Active Sensing. Despite the hints this thesis provides concerning the combined use of entropy and covariance for active sensing strategies under large initial uncertainties, it does not deal with active sensing. It seems like using POMDP techniques for active sensing under uncertainty are beyond the scope of current and "near future" computational capacities. Therefore, more "ad hoc" active sensing strategies using a combination of characteristics of the posterior density, are probably a good alternative.

Online Experimental Validation of the cube in corner assembly. Now that both the robot control software and the estimation software have reached a mature state, online and experimental validation of the setup seem to be feasible with a reasonable effort. This will probably increase the robustness of the algorithms with respect to variations in e.g. model parameter values such as the criteria used to decide if the current contact is stable or not.[1] This is also important for porting the approach to similar ACM problems. As recent research closed the gap between the output of the off-line planner and the controller component of the ACM system developed at our department, the major hurdle to take for the near future is to incorporate an ad hoc active sensing strategy that can deal with large uncertainties.

Furthermore, *global* sensors such as cameras or laser sensors provide global information about the environment in which robots operate. The data fusion of this complementary global information with the local information provided by encoders and force-torque sensors will allow to facilitate active sensing

[1] An alternative is to try to include these as hyper-parameters in the estimation process.

decisions considerably. Nowadays, cameras are cheap, widely available and their information can be acquired and processed in realtime.

References

Ackerson, G. and K. Fu (1970). On state estimation in switching environments. *IEEE Trans. Autom. Control 15*(1), 10–17.

Acklam, P. J. (1996). Monte Carlo methods in state space estimation. Master's thesis, Department of mathematics, University of Oslo.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. Petrov and F. Csaki (Eds.), *Proceedings of the Second Int. Symposium in Information Theory*, pp. 267–81. Budapest, Hungary: Akadémiai Kiadó.

Akashi, H. and H. Kumamoto (1977). Random sampling approach to state estimation in switching environments. *Automatica 13*, 429–434.

Alspach, D. and H. Sorenson (1972). Nonlinear Bayesian estimation using Gaussian sum approximations. *Trans. on Automatic Control 17*(4), 439–448.

Alvarado, P., P. Doerfler, and U. Canzler. LTI-Lib. `http://ltilib.sourceforge.net/`.

Anderson, B. and J. Moore (1979). *Optimal filtering*. Prentice-Hall, Englewood Cliffs, NJ.

Andrieu, C., M. Davy, and A. Doucet (2003). Efficient Particle Filtering for Jump Markov Systems. Application to Time-Varying Autoregressions. *IEEE Trans. Signal Processing 51*(7), 1762–1770.

Andrieu, C., N. de Freitas, and A. Doucet (1999). Sequential MCMC for Bayesian model selection. In *IEEE Higher Order Statistics Workshop*, Ceasarea, Israel.

Andrieu, C. and A. Doucet (2003). Online expectation-maximization type algorithms for parameter estimation in general state space models. In *Int. Conf. Acoustics, Speech and Signal Processing*, Volume 6, pp. 69–72.

Andrieu, C., A. Doucet, S. S. Singh, and V. B. Tadic (2004). Particle methods for change detection, system identification, and control. *Proc. of the*

IEEE 92(3), 423–438. Special Issue on: Sequential State Estimation: From Kalman Filters to Particle Filters.

Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Trans. Signal Processing 50*(2), 174–188.

Asada, H. (1993). Representation and learning of nonlinear compliance using neural nets. *IEEE Trans. Rob. Automation 9*(6), 863–867.

Aycard, O., J.-F. Mari, and R. Washington (2004). Learning to automatically detect features for mobile robots using second-order Hidden Markov Models. *Int. J. of Adv. Rob. Systems 1*(4), 231–245.

Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE 76*(8), 996–1005.

Bar-Shalom, Y. and T. Fortmann (1988). *Tracking and Data Association*. Mathematics in Science and Engineering. Academic Press.

Bar-Shalom, Y. and X. Li (1993). *Estimation and Tracking, Principles, Techniques, and Software*. Artech House.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics 41*, 164–171.

Bernardin, K., K. Ogawara, K. Ikeuchi, and R. Dillmann (2005). A sensor fusion approach for recognizing continuous human grasping sequences using Hidden Markov Models. *IEEE Trans. Rob 21*.

Bernardo, J. M. and A. F. M. Smith (1994). *Bayesian theory*. Chicester: Wiley.

Berzuini, C. and W. Gilks (2001). *Sequential Monte Carlo Methods in Practice*, Chapter RESAMPLE–MOVE Filtering with Cross–Model Jumps, pp. 117–138. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Bierman, G. (1974). Sequential square root filtering and smoothing of discrete linear systems. *Automatica 10*, 147–158.

Bierman, G. and C. Thornton (1977). Numerical comparison of Kalman filter algorithms. *Automatica 13*(1), 23–35.

Blom, H. (1984). An efficient filter for abruptly changing systems. In *Proc. of the 23rd IEEE Conf. on Decision and Control*, Las Vegas, pp. 656–658.

Blom, H. and Y. Bar-Shalom (1988). The interacting Multiple Model algorithm for systems with Markovian switching coefficients. *IEEE Trans. Autom. Control 33*(8), 780–783.

Bolic, M. (2004). *Architectures for Efficient Implementation of Particle Filters*. Ph. D. thesis, State University of New York at Stony Brook.

Bolic, M., P. Djuric, and S. Hong (2004). Resampling Algorithms and Architectures for Distributed Particle Filters. *IEEE Trans. Signal Processing*. to appear.

Bølviken, E. and G. Storvik (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Deterministic and Stochastic Particle Filters in State–Space Models, pp. 97–116. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Boost. Portable C++ libraries. `http://www.boost.org/`.

Boyen, X. and D. Koller (1998). Tractable inference for complex stochastic processes. In *Proc. of the 14th Annual Conference on Uncertainty in AI*, pp. 33–42.

Buckheit, J. and D. Donoho (1995). Wavelab and reproducible research. Technical report, Stanford University. `http://www-stat.stanford.edu/~donoho/Reports/1995/wavelab.pdf`.

Bucy, R. S. and H. Youssef (1974). Nonlinear filter representation via spline functions. In *5th Symposium on Nonlinear Estimation*, pp. 51–60.

Carpenter, J., P. Clifford, and P. Fearnhead (1999a). Building robust simulation-based filters for evolving data sets. Technical report, Department of Statistics, University of Oxford.

Carpenter, J., P. Clifford, and P. Fearnhead (1999b). An Improved Particle Filter for Non-linear Problems. *Radar, Sonar and Navigation, IEE Proc - 146*(1), 2–7.

Casell, G. and C. P. Robert (1996). Rao-Blackwellization of sampling schemes. *Biometrika 83*(1), 81–84.

Casella, G. and E. I. George (1992). Explaining the Gibbs Sampler. *The American Statistician 46*(3), 167–174.

Cassandra, A. R. (1998). *Exact and approximate algorithms for partially observable Markov decision processes*. Ph. D. thesis, U. Brown.

Charniak, E. (1991). Bayesian Networks without tears. *AI magazine 12*(4), 50–63.

Chib, S. and E. Greenberg (1995). Understanding the Metropolis–Hastings Algorithm. *The American Statistician 49*(4), 327–335.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika 89*(3), 539–551.

Chung, K. (1960). *Markov chains with stationary transition probabilities*. Berlin, Germany: Springer–Verlag.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Springer.

Cox, I. and J. Leonard (1991). Probabilistic data association for dynamic world modeling: A multiple hypothesis approach. In *Int. Conf. Advanced Robotics*, Pisa, Italy.

Cox, I. J. (1993). A review of statistical data association techniques for motion correspondence. *Int. J. of Computer Vision 10*(1), 53–667.

Crisan, D. and A. Doucet (2002). A Survey of Convergence Results on Particle Filtering Methods for Practitioners. *IEEE Trans. Signal Processing 50*(3), 736–746.

Crisan D., Del Moral, P. and T. Lyons (1999). Discrete filtering using branching and interacting particle systems. *Markov Proc. Rel. Fields 5*, 293–318.

Daum, F. E. (1988). New exact nonlinear filters. In J. C. Spall (Ed.), *Bayesian Analysis of Time Series and Dynamical Models*, pp. 265–292. Marcel Dekker.

Davies, R. (1994). Writing a matrix package in c++. In *The second annual object-oriented numerics conference*, pp. 207–213.

Davison, A. J. and N. Kita (2001). Sequential localisation and map-building for real-time computer vision and robotics. *RAS 36*, 171–183.

De Freitas, N., R. Dearden, F. Hutter, R. Morales-Menéndez, J. Mutch, and D. Poole (2004). Diagnosis by a Waiter and a Mars Explorer. *Proc. of the IEEE 92*(3), 455–468.

De Geeter, J., H. Van Brussel, J. De Schutter, and M. Decréton (1996). Recognising and locating objects with local sensors. In *Proc. of the IEEE Int. Conf. on Robotics and Aut.*, Minneapolis, Minnesota, pp. 3478–3483.

De Schutter, J., H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, and T. Lefebvre (1999). Estimating first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions. *Int. J. Robotics Research 18*(12), 1161–1184.

De Schutter, J., J. De Geeter, T. Lefebvre, and H. Bruyninckx (1999). Kalman filters: A tutorial. *J. A 40*(4), 52–59.

De Schutter, J. and H. Van Brussel (1988). Compliant Motion I, II. *Int. J. Robotics Research 7*(4), 3–33.

Dean, T. and K. Kanazawa (1989). A model for reasoning about persistence and causation. *Art. Intell. 93*(1–2), 1–27.

Debus, T., P. Dupont, and R. Howe (2002). Contact State Estimation using Multiple Model Estimation and Hidden Markov Models. In *Int. Symp. on Experimental Robotics*, Sant'Angelo d'Ischia, Italy.

Demeester, E., M. Nuttin, D. Vanhooydonck, and H. Van Brussel (2003). Assessing the user's intent using Bayes' rule: Application to wheelchair control. In *Proc. of ASER 2003*, Bardolino, Italy, pp. 117–124.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *J. of the Royal Statistical Society, Series B 39*, 1–38.

Devroye, L. (1985). *Non-Uniform Random Variate Generation*. New York: Springer-Verlag.

Diaconis, P. and D. Ylvisaker (1979). Conjugate priors for exponential families. *Annals of Statistics 7 7*, 269–281.

Dissanayake, M., P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Rob. Automation 17*(3), 229–241.

Doucet, A. (1997). *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models*. Ph. D. thesis, Univ. Paris-Sud, Orsay. in french.

Doucet, A. (1998). On Sequential Simulation-Based Methods for Bayesian Filtering. Technical Report CUED/F-INFENG/TR.310, Signal Processing Group, Dept. of Engineering, University of Cambridge.

Doucet, A., N. de Freitas, and N. Gordon (Eds.) (2001). *Sequential Monte Carlo Methods in Practice*. Statistics for engineering and information science. Springer.

Doucet, A., N. de Freitas, and E. Punskaya. The sequential monte carlo methods homepage. `http://www-sigproc.eng.cam.ac.uk/smc/index.html`.

Doucet, A., S. Godsill, and C. Andrieu (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing 10*(3), 197–208.

Doucet, A., N. J. Gordon, and V. Krishnamurthy (2001). Particle Filters for State Estimation of Jump Markov Linear Systems. *IEEE Trans. Signal Processing 49*(3), 613–624.

Doucet, A. and V. B. Tadic (2003). Parameter Estimation in General State-Space Models using Particle Methods. *Annals of the Institute of Statistical Mathematics 55*(2), 409–422.

Duda, R. O. and P. E. Hart (1973). *Pattern classification and scene analysis*. New York, NY: Wiley.

Dugad, R. and U. Desai (1996). A tutorial on Hidden Markov Models. Technical Report SPANN-96.1, Indian institute of Technology, dept. of electrical engineering, Signal Processing and Artificial Neural Networks Laboratory, Bombay, Powai, Mumbai 400 076 India. `http://vision.ai.uiuc.edu/dugad/newhmmtut.ps.gz`.

Farook, M. and S. Bruder (1990). Information type filters for tracking a maneuvering target. *Trans. on Aerospace and Electronic Systems 26*(3), 441–454.

Fearnhead, P. Using Random Quasi-Monte-Carlo within Particle Filters, with Application to Financial Time Series. *J. of Computational Graphical Statistics*. To appear.

Fearnhead, P. (1998). *Sequential Monte Carlo methods in filter theory*. Ph. D. thesis, Merton College, University of Oxford.

Ferguson, J. D. (1980). Variable duration models for speech. In *Proceedings of the Symposium on the Application of Hidden Markov Models to Text and Speech*, pp. 143–179.

Forney, G. D. J. (1973). The Viterbi algoritm. In *Proc. IEEE*, Volume 61, pp. 263–278.

Fortmann, T. E., Y. Bar-Shalom, and M. Scheffe (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Oceanic Eng. 8*, 173–184.

Fox, D. (2001). KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems 14*. MIT Press.

Fox, D. (2003). Adapting the Sample Size in Particle Filters Through KLD-Sampling. *Int. J. Robotics Research 22*(12), 985–1003.

Fox, D., W. Burgard, F. Dellaert, and S. Thrun (1999). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, Orlando, FL.

Fox, D., W. Burgard, and S. Thrun (1998). Active Markov localization for mobile robots. *Rob. Auton. Systems 25*, 195–207.

Fox, D., W. Burgard, and S. Thrun (1999). Markov localization for mobile robots in dynamic environments. *J. of Artificial Intelligence Research 11*, 391–427.

Fox, D., S. Thrun, W. Burgard, and F. Dellaert (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Particle Filters for Mobile Robot Localization, pp. 401–428. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Gadeyne, K. (2001a). BFL API documentation. `http://people.mech.kuleuven.ac.be/~kgadeyne/software/actsens/software/filter/doc/html/index.html`.

Gadeyne, K. (2001b). BFL: Bayesian Filtering Library. `http://people.mech.kuleuven.ac.be/~kgadeyne/bfl.html`.

Gadeyne, K. and H. Bruyninckx (2001). Markov techniques for object localisation with force-controlled robots. In *Int. Conf. Advanced Robotics*, Budapest, Hungary, pp. 91–96.

Gadeyne, K., T. Lefebvre, and H. Bruyninckx (2005). Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation. *Int. J. Robotics Research 24*(8), 615–630.

Gelfland, A. E. and A. F. M. Smith (1990). Sampling-Based Approaches to Calculating Marginal Densities. *J. Amer. Statistical Association 85*(410), 398–409.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell. 6*, 721–741.

Geweke, J. (1989). Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica 57*(6), 1317–1339.

Ghahramani, Z. and G. Hinton (1998). Variational learning for switching state-space models. *Neural Computation 12*(4), 963–996.

Gilks, W. R. and C. Berzuini (2001). Following a moving target–Monte Carlo inference for dynamic Bayesian models. *J. of the Royal Statistical Society 63*(Part 1), 127–146.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (Eds.) (1996). *Markov Chain Monte Carlo in Practice* (First ed.). London: Chapman & Hall.

Godsill, S. and T. Clapp (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Improvement Strategies for Monte Carlo Particle Filters, pp. 139–158. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Goncalves, R., P. Moraes, J. Cardoso, D. Wolf, M. Fernandes, R. Romero, and E. Marques (2003). ARCHITECT-R: a system for reconfigurable robots design. In A. Press (Ed.), *ACM Symposium on Applied Computing*, pp. 679–683.

Gordon, N., D. Salmond, and C. Ewing (1995). Bayesian state estimation for tracking and guidance using the bootstrap filter. *J. Guid. Cont. Dynamics 18*(6), 1434–1443.

Gordon, N., D. J. Salmond, and A. F. M. Smith (1993). Novel approach to nonlinear/non-Gaussian state estimation. *IEE Proceedings-F 140*(2), 107–113.

gR. gR: gRaphical models in R. `http://www.r-project.org/gR/`.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*, 711–732.

Hamilton, J. (1990). Analysis of time series subject to changes in regime. *J. of Econometrics 45*, 39–70.

Hammersley, J. and D. Handscomb (1964). *Monte Carlo Methods*. Monographs on Applied Probability and Statistics. Chapman and Hall.

Handschin, J. and D. Mayne (1969). Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control 9*, 547–559.

Hannaford, B. and P. Lee (1991). Hidden Markov Model analysis of force/torque information in telemanipulation. *Int. J. Robotics Research 10*(5), 528–539.

Harrison, P. and C. Stevens (1976). Bayesian forecasting. *J. R. Statist. Soc. B 38*(3), 205–247.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika 57*, 97–107.

Herbrich, R. (2002). *Learning Kernel Classifiers*. MIT Press.

Hirukawa, H., Y. Papegay, and T. Matsui (1994). A motion planning algorithm for convex polyhedra in contact under translation and rotation. In *Int. Conf. on Robotics and Aut.*, pp. 3020–3027.

Hovland, G. E. and B. J. McCarragher (1998). Hidden Markov Models as a Process Monitor in Robotic Assembly. *Int. J. Robotics Research 17*(2), 153–168.

Howard, A. and N. Roy (2003). The robotics data set repository (radish). `http://radish.sourceforge.net/`.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, Massachusetts: The MIT Press.

Hürzeler, M. and H. Künsch (1995). Monte Carlo approximations for general state-space models. Technical Report 73, Seminar für Statistik, Eidgenössische Technische Hochschule.

Iba, Y. (2001). Population monte carlo algorithms. *Trans. of the Jap. Soc. for AI 16*(2), 279–286.

Intel. Open Probabilistic Network library (OpenPNL). `http://www.intel.com/research/mrl/pnl/`.

Isard, M. and A. Blake (1998). Condensation—conditional density propagation for visual tracking. *Int. J. Computer Vision 29*(1), 5–28.

Jaakkola, T. and M. Jordan (1999). Variational probabilistic inference and the QMR-DT network. *J. AI Res. 10*, 291–322.

Jaakkola, T. and M. Jordan (2000). Bayesian parameter estimation via variational methods. *Stat. and Comp. 10*(1), 25–37.

Jaynes, E. T. (1996). Probability theory: The logic of science. Unfinished manuscript, `http://bayes.wustl.edu/etj`.

Jeffreys, H. (1939). *Theory of Probability*. Clarendon Press. 2nd edition, 1948; 3rd edition, 1961. Reprinted by Oxford University Press, 1998.

Jordan, M. I. (Ed.) (1999). *Learning in Graphical Models*. Adaptive Computation and Machine Learning. London, England: MIT Press. ISBN 0262600323.

Julier, S. and J. Uhlmann (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE 92*(3), 401–422. Special Issue on: Sequential State Estimation: From Kalman Filters to Particle Filters.

Julier, S. J. and J. K. Uhlmann (1997). A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL. SPIE.

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence 101*, 99–134.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng. 82*, 34–45.

Kalos, M. H. and P. A. Whitlock (1986). *Monte Carlo methods*, Volume I: Basics of *Wiley-intersience publications*. New York: Wiley.

Kaminsky, P., A. Bryson, and S. Schmidt (1971). Discrete Square Root Filtering, a survey of current techniques. *IEEE Trans. Autom. Control 16*(6), 727–736.

Kass, R. and A. Raftery (1995). Bayes factors. *J. Amer. Statistical Association 90*(430), 773–795.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time series (with discussion). *J. Amer. Statistical Association 82*(400), 1032–1063.

Kitagawa, G. (1993). A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state-space models. In *Proc. of the 2nd US-Japan Joint Seminar on Statistical Time Series Analysis*, Honolulu, pp. 110–131.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. of Computational and Graphical Statistics 5*(1), 1–25.

Kitagawa, G. (1998). A self-organising state-space model. *J. Amer. Statistical Association 93*(443), 1203–1215.

Kröse, G. J. A. and R. Bunschoten (1999). Probabilistic localization by appearance models and active vision. In *IEEE conference on Robotics and Aut., Detroit*.

Kullback, S. and R. Leibler (1951). On information and sufficiency. *Annals of mathematical Statistics 22*, 79–86.

Lauritzen, S. L. and D. J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *J. R. Statist. Soc. B 50*(2), 157–224. Reprinted in (Shafer and Pearl 1990, p. 415).

Lee, D. S. and N. K. K. Chia (2002). A particle algorithm for sequential Bayesian parameter estimation and model selection. *IEEE Trans. Signal Processing 50*(2), 326–336.

Lefebvre, T. (2003). *Contact modelling, parameter identification and task planning for autonomous compliant motion using elementary contacts*. Ph. D. thesis, Dept. Mechanical Engineering, Katholieke Universiteit Leuven.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2003). Polyhedral contact formation modeling and identification for autonomous compliant motion. *IEEE Trans. Rob. Automation 19*(1), 26–41.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2004a). Exact nonlinear Bayesian parameter estimation for autonomous compliant motion. *Advanced Robotics 18*(8), 787–800.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2004b). Kalman Filters for nonlinear systems: a comparison of performance. *International Journal of Control 77*(7), 639–653.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005a). Polyhedral contact formation identification for autonomous compliant motion: Exact nonlinear Bayesian filtering. *IEEE Trans. Rob. Automation 21*(1), 124–129.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005b). Task planning with active sensing for autonomous compliant motion. *Int. J. Robotics Research 24*(1), 61–82.

Lefebvre, T., K. Gadeyne, H. Bruyninckx, and J. De Schutter (2003). Exact Bayesian inference for a class of nonlinear systems with application to

robotic assembly. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, D. H. A. P. Dawid, A. F. M. Smith, and M. West (Eds.), *Bayesian Statistics 7*, pp. 587–596.

LeGland, F. and N. Oudjane (2004). Stability and Uniform Approximation of Nonlinear Filters using the Hilbert Metric, and Application to Particle Filters. *The Annals of Applied Probability 14*(1), 144–187.

Lenser, S. and M. Veloso (2000). Sensor resetting localization for poorly modelled mobile robots. In *Int. Conf. Robotics and Automation*, San Francisco, CA.

Levinson, S. E. (1986a). Continuously Variable Duration Hidden Markov Models for speech analysis. In *Int. Conf. on Acoustics, Speech, and Signal Processing*, Volume 2, pp. 1241–1244. AT&T Bell Lab.

Levinson, S. E. (1986b). Continuously Variable Duration Hidden Markov Models for speech recognition. *Computer, Speech and Language 1*, 29–45.

Lindley, D. V. (1972). *Bayesian statistics: a review*. SIAM.

Liu, J. and R. Chen (1995). Blind deconvolution via sequential imputations. *J. Amer. Statistical Association 6*(90), 567–576.

Liu, J. S. (1996). Metropolised independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing 6*, 113–119.

Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. Springer series in Statistics. Springer.

Liu, J. S. and R. Chen (1998). Sequential Monte Carlo methods for dynamic systems. *J. Amer. Statistical Association 93*(443), 1032–1044.

Liu, J. S., R. Chen, and W. H. Wong (1998). Rejection Control and Sequential Importance Sampling. *J. Amer. Statistical Association 93*(443), 1022–1031.

Liu, J. S. and M. West (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Combined Parameter and State Estimation in Simulation-Based Filtering, pp. 197–223. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

MacEachern, S. N., M. A. Clyde, and J. S. Liu (1999). Sequential importance sampling for nonparametric Bayes models: The next generation. *Canadian J. of Statistics 27*(2), 251–267.

MacKay, D. J. C. (1995). Probable networks and plausible predictions—A review of practical Bayesian methods for supervised neural networks. In *Network*.

MacKay, D. J. C. (2003). Information theory, inference, and learning algorithms.

Mantegazza, P. RTAI: the Real-Time Applications Interface. `http://www.rtai.org`.

Marsaglia, G. and A. Zaman (1993). The KISS generator. Technical report, Dept. of Statistics, Florida State University.

Martin, A. D., K. M. Quinn, and D. Pemstein. Scythe statistical library. `http://scythe.wustl.edu/`.

Maybeck, P. S. (1982). *Stochastic models, estimation, and control. Vol. 2.* Number 141-2 in Mathematics in science and engineering. Orlando, FL: Academic Press. Republished by Navtech Press, Arlington, 1994.

McGinnity, S. and G. W. Irwin (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Manoeuvring Target Tracking Using a Multiple-Model Bootstrap Filter, pp. 479–497. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Meeussen, W., J. De Schutter, H. Bruyninckx, J. Xiao, and E. Staffetti (2005). Integration of planning and execution in force controlled compliant motion. In *Proc. IEEE/RSJ Int. Conf. Int. Robots and Systems*, Edmonton, Canada.

Mehra, R. K. and J. Peschon (1971). An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica 7*, 637–640.

Mekhnacha, K., E. Mazer, and P. Bessière (2001). The design and implementation of a Bayesian CAD modeler for robotic applications. *Advanced Robotics, the International Journal of the Robotics Society of Japan 15*(1), 45–70.

Metropolis, N. and S. Ulam (1949). The Monte Carlo Method. *J. Amer. Statistical Association 44*, 335–341.

Metropolis, N. C., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equations of state calculations by fast computing machine. *J. of Chemical Physics 21*, 1087–1091.

Mihaylova, L., T. Lefebvre, E. Staffetti, H. Bruyninckx, and J. De Schutter (2002). Contact transitions tracking during force-controlled compliant motion using an interacting multiple model estimator. *Information & Security 9*, 114–129.

Mimura, N. and Y. Funahashi (1994). Parameter identification of contact conditions by active force sensing. In *Int. Conf. Robotics and Automation*, San Diego, CA, pp. 2645–2650.

Minka, T. (2001). *A family of algorithms for approximate Bayesian inference*. Ph. D. thesis, MIT.

Minka, T. (2004). Bayesian inference in dynamic models – an overview. http://www.stat.cmu.edu/~minka/dynamic.html.

Montemerlo, M., S. Thrun, D. Koller, and B. Wegbreit (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Nat. Conf. on Artificial Intelligence*, pp. 593–598.

Montemerlo, M., S. Thrun, D. Koller, and B. Wegbreit (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Int. Jnt. Conf. Art. Intelligence*, Acapulco, Mexico.

Montemerlo, M., W. Whittaker, and S. Thrun (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Int. Conf. Robotics and Automation*, Washington DC, U.S.A.

Murphy, K. P. (1998). Switching Kalman Filters. Technical report, U.C. Berkeley.

Murphy, K. P. (2000). A Survey of POMDP Solution Techniques. Technical report, U.C. Berkeley.

Murphy, K. P. (2001). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*.

Murphy, K. P. (2002a). Dynamic Bayesian Networks. To appear in *Probabilistic Graphical Models*, M. Jordan.

Murphy, K. P. (2002b). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph. D. thesis, UC Berkeley, Computer Science Division.

Murphy, K. P. (2004a). The Bayes Net Toolbox for Matlab. http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html. checked 07/05/2005.

Murphy, K. P. (2004b). Software packages for graphical models / Bayesian Networks. http://www.ai.mit.edu/~murphyk/Bayes/bnsoft.html. Last checked 07/05/2005.

Murphy, K. P. and S. Russell (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Rao-Blackwellised particle filtering for dynamic Bayesian networks, pp. 499–516. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Musser, D. R. and A. Saini (1996). *STL Tutorial and Reference Guide. C++ Programming with the Standard Template Library*. Addison-Wesley.

Musso, C., N. Oudjane, and F. LeGland (2001). *Sequential Monte Carlo Methods in Practice*, Chapter Improving Regularised Particle Filters, pp. 247–269. In Doucet et al. Doucet, de Freitas, and Gordon (2001).

Neal, R. M. (1993). Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Department of Computer Science.

Neal, R. M. (1998). Annealed Importance Sampling. Technical Report 9805, Dept. of Statistics and dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Neal, R. M. (2003). Slice Sampling. *Annals of Statistics 31*(3), 705–767.

Neira, J. and J. D. Tardós (2001). Data Association in stochastic mapping using the Joint Compatibility Test. *IEEE Trans. Rob. Automation 17*(6), 890–897.

Open Management Group. CORBA: Common Object Request Broker Architecture. `http://www.corba.org/`.

Open Source Initiative. The Open Source Page. `http://www.opensource.org/`.

Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In *Int. Jnt. Conf. Art. Intelligence*, Acapulco, Mexico, pp. 1157–1164.

Pitt, M. and N. Shephard (1999). Filtering via simulation: Auxiliary particle filters. *J. Amer. Statistical Association 94*(446), 590–599.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286.

Rabiner, L. R. and B. H. Juang (1986). An introduction to Hidden Markov Models. *IEEE ASSP Magazine 3*(1), 4–16.

Raibert, M. and J. J. Craig (1981). Hybrid position/force control of manipulators. *Trans. ASME J. Dyn. Systems Meas. Control 102*, 126–133.

Reid, D. (1979). An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control 24*(6), 843–854.

Ripley, B. D. (1987). *Stochastic Simulation*. John Wiley and Sons.

Robert, C. P. and G. Casella (1999). *Monte Carlo Statistical Methods*. Springer.

Rubin, D. B. (1988). *Bayesian Statistics 3*, Chapter Using the SIR algorithm to simulate posterior distributions, pp. 395–402. Oxford University Press. Using the SIR algorithm to simulate posterior distributions.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. Wiley series in probability and mathematical statistics. Wiley.

Rutgeerts, J. e. a. (2005). A demonstration tool with Kalman Filter data processing for robot programming by human demonstration. In *Proc. IEEE/RSJ Int. Conf. Int. Robots and Systems*, Edmonton, Canada.

Schulz, D. and W. Burgard (2001). Probabilistic state estimation of dynamic objects with a moving mobile robot. *Robotics and Autonomous Systems 34*(2-3), 107–115.

Schulz, D., W. Burgard, and D. Fox (2003). People tracking with mobile robots using sample-based joint probabilistic data association filters. *Int. J. Robotics Research 22*(2), 99–116.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics 6*, 461–464.

Shafer, G. and J. Pearl (Eds.) (1990). *Readings in Uncertain Reasoning*. San Mateo, CA: Morgan Kaufmann.

Shumway, R. H. and D. S. Stoffer (1982). An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis 3*(4), 253–264.

Shumway, R. H. and D. S. Stoffer (1992). Correction: Dynamic linear models with switching. *J. Amer. Statistical Association 87*(419), 913.

Silverman, B. (1986). *Density estimation for statistics and data analysis*. London: Chapman and Hall.

Simmons, R. and S. Koenig (1995). Probabilistic robot navigation in partially observable environments. In *Int. Jnt. Conf. Art. Intelligence*, Montreal, Canada, pp. 1080–1087.

Skubic, M. and R. Volz (2000). Identifying single-ended contact formations from force sensor patterns. *IEEE Trans. Rob. Automation 16*(5), 597–603.

Slaets, P., T. Lefebvre, H. Bruyninckx, and J. De Schutter (2004). Construction of a Geometrical 3-D Model from Sensor Measurements Collected during Compliant Motion. *IEEE Trans. Rob*. rejected.

Slaets, P., J. Rutgeerts, K. Gadeyne, T. Lefebvre, H. Bruyninckx, and J. De Schutter (2004). Construction of a Geometric 3-D Model from Sensor Measurements Collected during Compliant Motion. In *Int. Symp. on Experimental Robotics*, Singapore, Australia. in press.

Smith, A. F. M. and A. E. Gelfland (1992). Bayesian Statistics Without Tears: A Sampling–Resampling Perspective. *The American Statistician 46*(2), 84–88.

Smith, P. and G. Buechler (1975). A branching algorithm for discriminating and tracking multiple objects. *IEEE Trans. Autom. Control 20*, 101–104.

Smyth, P. (1994). Hidden Markov Models for fault detection in dynamic systems. *Pattern Recognition 27*(1), 149–164.

Soetens, P. and H. Bruyninckx (2005). Realtime hybrid task-based control for robots and machine tools. In *Int. Conf. Robotics and Automation*, Barcelona, Spain, pp. 260–265.

Sondik, E. J. (1971). *The Optimal Control of Partially Observable Markov Processes*. Ph. D. thesis, Stanford University, Stanford, California.

Stallman, R. M. GNU Public License. `http://www.fsf.org/copyleft/gpl.html`.

Stevens, M. (2003). Bayes++: Open Source Bayesian filtering classes. `http://bayesclasses.sourceforge.net/`.

Tanizaki, H. (1993). *Nonlinear filters*, Volume 400 of *Lecture notes in Economics and Mathematical systems*. Springer–Verlag.

Tanner, M. (1992). *Tools for Statistical Inference: Observed Data and Data Augmentation Methods* (Second ed.), Volume 67 of *Lecture notes in Statistics*. Springer–Verlag.

Tanner, M. (1996). *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions* (Third ed.). Lecture notes in Statistics. Springer–Verlag.

Thrun, S. (1998a). Bayesian landmark learning for mobile robot localization. *Machine Learning 33*(1), 41–76.

Thrun, S. (1998b). A framework for programming embedded systems: Initial design and results. Technical Report CMUCS-98-142, Carnegie Mellon.

Thrun, S. (2000). Towards programming tools for robots that integrate probabilistic computation and learning. In *Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 306–312.

Thrun, S. (2003). *Exploring Artificial Intelligence in the New Millenium*, Chapter Robotic Mapping: A Survey. Morgan Kaufmann.

Thrun, S., Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte (2004a). Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Int. J. Robotics Research 23*(7), 693–716.

Thrun, S., Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte (2004b). Simulataneous localization and mapping with sparse extended information filters. *Int. J. Robotics Research 23*(7–8), 693–716.

Thrun, S., C. Martin, Y. Liu, D. Hähnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard (2004). A Real-Time Expectation-

Maximization Algorithm for Acquiring Multiplanar Maps of Indoor Environments with Mobile Robots. *IEEE Trans. Rob. Automation 20*(3), 433–442.

Ueda, N. and R. Nakano (1998). Deterministic annealing EM algorithm. *Neural Networks 11*, 271–282.

van der Merwe, R. ReBEL: Recursive Bayesian Estimation Library. `http://choosh.ece.ogi.edu/rebel/`.

van der Merwe, R., A. Doucet, N. de Freytas, and E. Wan (2000). The Unscented Particle Filter. Technical Report CUED/F-INFENG/TR 380, Cambridge university engineering department, Cambridge CB2 1PZ, England.

van der Merwe, R. and E. Wan (2003). Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hong Kong.

Vanhooydonck, D., E. Demeester, M. Nuttin, and H. Van Brussel (2003). Shared control for intelligent wheelchairs: an implicit estimation of the user intention. In *Proc. of ASER 2003*, Bardolino, Italy, pp. 176–182.

Vermaak, J., S. Godsill, and P. Pérez (2005). Monte Carlo Filtering for Multi-Target Tracking and Data Association. *IEEE Trans. on Aerospace and Electronic Systems*. To appear.

Verman, V., G. Gordon, R. Simmons, and S. Thrun (2004). Real-Time Fault Diagnosis. *IEEE Rob. Automation Mag. 11*(2), 56–66.

Wan, E. and A. Nelson (1997). Dual kalman filtering methods for nonlinear prediction, estimation, and smoothing. In J. Mozer and Petsche (Eds.), *In Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference , NIPS-9*.

Watkin, T. (1993). Optimal learning with a neural network. *Europhysics Letters* (21), 871–876.

West, M. (1993a). Approximating posterior distributions by mixtures. *J. R. Statist. Soc. B 55*(2), 409–422.

West, M. (1993b). Mixture models, Monte Carlo, Bayesian updating and dynamic models. In J. Newton (Ed.), *Comp. Science and Stat.: Proc. of the 24th symp. on the Interface*, Virginia, pp. 325–333. Interface Foundation of North America.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica 12*, 601–611.

Willsky, A. S. and H. L. Jones (1976). A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Trans. Autom. Control*, 108–112.

Xiao, J. and X. Ji (2000). A Divide-And-Merge Approach to Automatic Generation of Contact States and Planning of Contact Motions. In *Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 750–756.

Xiao, J. and X. Ji (2001). On automatic generation of high-level contact state space. *Int. J. Robotics Research 20*(7), 584–606.

Zhang, N. and D. Poole (1996). Exploiting causal independence in Bayesian Network inference. *J. AI Res.*, 301–328.

Appendix A

The Expectation Maximization algorithm

If we take the logarithm, multiply the left and right part of equation (5.24) with the posterior state estimate given a fixed parameter value $\boldsymbol{\theta}^t$,[1] and integrate over the states,[2] we obtain

$$\begin{aligned} &\int log\left(P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta})\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k} \\ &= \int log\left(\frac{P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})}{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}, \boldsymbol{z}_{1:k})}\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k}, \end{aligned} \tag{A.1}$$

and consequently

$$\begin{aligned} log\left(P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta})\right) = \quad & E_{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})}\left[log\left(P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})\right)\right] \\ & - E_{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})}\left[log\left(P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}, \boldsymbol{z}_{1:k})\right)\right]. \end{aligned} \tag{A.2}$$

Remember we're looking to for values $\boldsymbol{\theta}$ that maximise the latter observed likelihood iteratively. Define

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = E_{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})}\left[log\left(P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})\right)\right]. \tag{A.3}$$

Now, we will proof that if we find a value $\boldsymbol{\theta}^{t+1}$ for which $Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t) > Q(\boldsymbol{\theta}^t, \boldsymbol{\theta}^t)$, this implies that $P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{t+1}) > P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{t+1})$. In that way, we're

[1] The *superscript* ^{t} denotes an particular value of $\boldsymbol{\theta}$ during the iteration process, and has nothing to do with the subscript k that relates to the time of the filter!

[2] This corresponds to calculating the expected value with respect to the state posterior $P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^t, \boldsymbol{z}_{1:k})$.

closer to the local maximum of the observed data likelihood function.

$$\begin{aligned}
&P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{t+1}) - P(\boldsymbol{z}_{1:k}|\boldsymbol{\theta}^{t}) \\
&= Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^{t}) - \int log\left(P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t+1}, \boldsymbol{z}_{1:k})\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k} \\
&\quad - \left(Q(\boldsymbol{\theta}^{t}, \boldsymbol{\theta}^{t}) - \int log\left(P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k})\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k}\right) \\
&= Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^{t}) - Q(\boldsymbol{\theta}^{t}, \boldsymbol{\theta}^{t}) \\
&\quad - \int log\left(\frac{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t+1}, \boldsymbol{z}_{1:k})}{P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k})}\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k}.
\end{aligned} \tag{A.4}$$

Now, since $log(a) \leq a - 1$, the resulting integral is less than zero. Therefore if $Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^{t}) > Q(\boldsymbol{\theta}^{t}, \boldsymbol{\theta}^{t})$, $\mathcal{L}^{o}(\boldsymbol{\theta}^{k+1}|\boldsymbol{z}_{1:k}) > \mathcal{L}^{o}(\boldsymbol{\theta}^{k}|\boldsymbol{z}_{1:k})$.

All that remains is to find a value $\boldsymbol{\theta}^{t+1}$ so that $Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^{t}) > Q(\boldsymbol{\theta}^{t}, \boldsymbol{\theta}^{t})$ or (even better)

$$\boldsymbol{\theta}^{t+1} = ArgMax_{\boldsymbol{\theta}}\ Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t}). \tag{A.5}$$

The first method converges more slowly and is sometimes referred to as *generalized* EM (Dempster, Laird, and Rubin 1977). The calculation of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t})$ involves running a filter to with the current parameter estimate $\boldsymbol{\theta}^{t}$. So during *each* iteration step, a filter is run over all available measurements. Since

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t}) = \int log\left(P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta})\right) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k}) d\boldsymbol{x}_{1:k}. \tag{A.6}$$

The filter provides an estimate of $P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}^{t}, \boldsymbol{z}_{1:k})$. The first term of this integral can be factorized as

$$\begin{aligned}
&P(\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}|\boldsymbol{\theta}) \\
&= P(\boldsymbol{z}_{1:k}|\boldsymbol{x}_{1:k}, \boldsymbol{\theta}) P(\boldsymbol{x}_{1:k}|\boldsymbol{\theta}) \\
&=^{1} \prod_{j=1}^{k} \left[P(\boldsymbol{z}_{j}|\boldsymbol{x}_{j}, \boldsymbol{\theta}) P(\boldsymbol{x}_{j}|\boldsymbol{x}_{j-1}, \boldsymbol{\theta})\right] P(\boldsymbol{x}_{0}|\boldsymbol{\theta}).
\end{aligned} \tag{A.7}$$

Where [1] uses the Markov assumption. So the first term of integral (A.6) can be rewritten as the product of a system the measurement model, written as a function of $\boldsymbol{x}$ and $\boldsymbol{\theta}$.

The resulting integral can be calculated analytically (e.g. in the case of a KF), or with Monte Carlo methods (Tanner 1992). The combination of a Kalman Filter and EM is sometimes referred as *Dual Kalman Filtering* (Wan and Nelson 1997).

Curriculum Vitae

Personal data

Klaas Gadeyne
Date and place of birth: 30 September 1977, Roeselare, Belgium
Nationality: Belgian

Work address:
Flanders' MECHATRONICS Technology Centre,
Celestijnenlaan 300D, B-3001 Leuven (Heverlee), Belgium
Tel: (+32) 16 32 80 61, Fax: (+32) 16 32 80 64

Email: klaas.gadeyne@gmail.com
Homepage: http://people.mech.kuleuven.be/∼kgadeyne/

Education

- **2005-**: Senior Project Engineer at the Flanders' MECHATRONICS Technology Centre, Heverlee, Belgium.
- **2000-2005**: Ph.D. student at the Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

  My research is situated in the area of autonomous compliant motion for force controlled robot tasks, such as deburring and assembly. The aim of this research is to make industrial robots work autonomously in less structured environments where they have to deal with inaccurately positioned tools and work pieces.

- **1995-2000**: **Master in mechanical engineering** (option mechatronics and machine design) at K.U.Leuven, Belgium.

  1997-1998: ECTS (European Credit Transfer System) study exchange with the Université de Technologie de Compiègne, France.

Nederlandse Samenvatting

Nederlandse Samenvatting

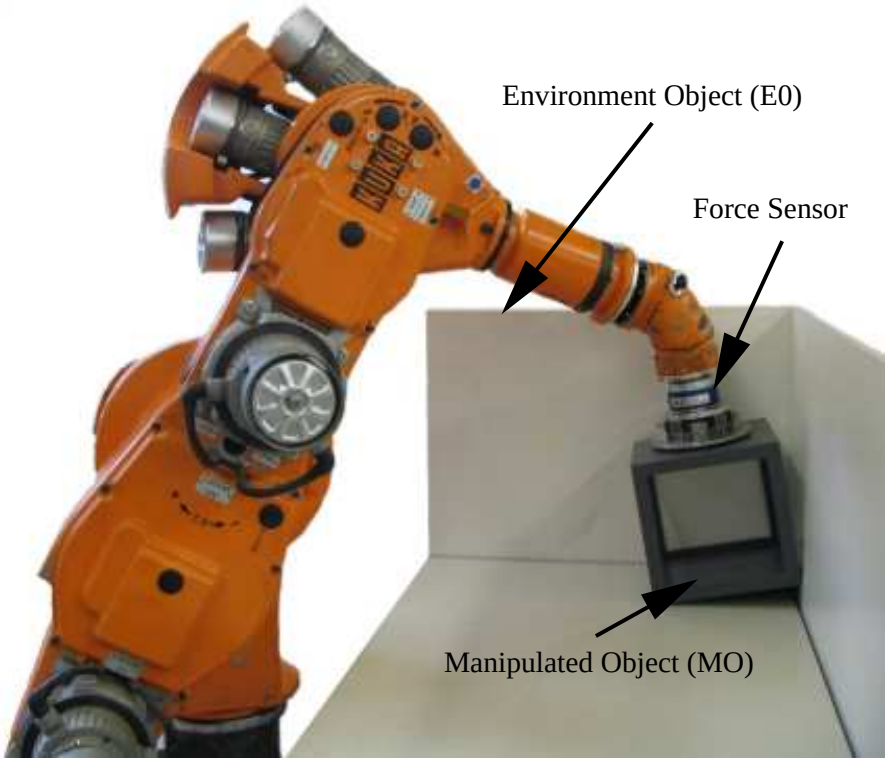
1 Inleiding

1.1 Situering

Dit doctoraat gaat over het autonoom maken van robots bij het uitvoeren van krachtgecontroleerde taken, waarbij de eindeffector of een werktuig dat vastzit op de eindeffector contact maakt met de omgeving (Eng. Autonomous compliant motion, ACM). Op die manier kunnen robots taken uitvoeren in niet (volledig) gestructureerde omgevingen.

De applicaties dat gebruik maakt van autonoom uitgevoerde krachtgecontroleerde taken in ongestructureerde omgevingen zijn legio: De ultieme huis-, tuin- en keuken robot moet in staat zijn om deuren te open en te sluiten, of die nu op een kier staan of dicht zijn. Bovendien moet hij (of zij) ook in staat zijn om de tafel af te ruimen, ongeacht van de rommel die wij er 's morgen op hebben achtergelaten. Industriële toepassingen vinden we in assemblagetaken en het bewerken van werkstukken (ontbramen, frezen, slijpen ...), zonder gebruik te moeten maken van dure bevestigingen en veel tijd te verliezen met alles off-line op te meten.

Deze gebruikt 1 concreet voorbeeld van ACM: een assemblagetaak waarbij een krachtgecontroleerde robot een kubus (het gemanipuleerde object, Eng. manipulated object MO) "zo snel mogelijk" in een hoek (het omgevingsobject, Eng. environment object EO) moet plaatsen (zie figuur 1). Hierbij weet de robot niet precies hoe hij de kubus vastgenomen heeft, en waar de hoek zich precies in zijn omgeving bevindt. Deze onbekenden noemt met de geometrische parameters. Om zijn taak tot een goed einde te brengen, maakt de robot gebruik van encoders en een krachtsensor. De assemblage verloopt als een opeenvolging van een aantal *contactformaties*, bv. een punt-vlak contact, een lijn-vlak contact of een 2-vlak-vlak contact. Elke discrete contactformatie zorgt voor een ander verband tussen de ongekende geometrische parameters en de metingen. Daarom is het nodig om de huidige contactformatie te (her)kennen. In het hybride krachtcontrole paradigma, is de herkenning van de contactformatie ook nodig om een geschikt controlealgoritme te kiezen.



Figuur 1: Autonome uitvoering van een kubus-in-hoekassemblage. Een Kuka 361 seriële robot, uitgerust met een krachtsensor, plaatst autonoom een kubus (het gemanipuleerde object MO) in een hoek, gevormd door drie loodrecht op elkaar staande wanden (het omgevingsobject EO). De positie en oriëntatie van de kubus t.o.v. de eindeffector van de robot en de positie en oriëntatie van de hoek t.o.v. een vast wereldassenstelsen zijn ongekend. Deze onbekenden worden vaak omschreven als de *geometrische parameters* en zijn continu van aard. De krachtsensor en de encoders leveren metingen die gebruikt worden om de waarden van deze parameters te schatten.

Tot voor deze thesis gebeurde de herkenning van contactformaties door gebruik te maken van een consistentietest, die naging hoe groot de kans dat de huidige metingen afkomstig zijn van de veronderstelde contactformatie. Als de onzekerheid op de geometrische parameters klein is, kunnen we de contactformatiesequentie voorspellen aan de hand van een off-line gegenereerd taakplan. Van zodra de metingen niet meer afkomstig zijn van de huidige contactformatie, weten we dat ze afkomstig zijn van de volgende contactformatie uit het taakplan. Deze aanpak faalt echter als er, door grote initiële onzekerheden op de geometrische parameters, dubbelzinnigheden mogelijk zijn in de contactformatiesequentie. Een eenvoudig voorbeeld hiervan is te zien in figuur 6.3 op pagina 106: Als de initiële onzekerheid op de geometrische parameters van het EO groot zijn, kunnen we wel detecteren dat we te maken hebben met een punt-vlak contact, maar niet precies met welk vlak van het EO de kubus in contact is.

Dit werk ontwikkelt een hybride (gedeeltelijk continu, gedeeltelijk discreet) estimatie-algoritme, dat een expliciet stochastisch transitiemodel gebruikt voor contactformatietransitiepredictie en waar zowel positie- als krachtinformatie gebruikt worden om contactformatietransities te detecteren. We verklaren waarom Kalman Filter, die vroeger gebruikt werden voor de estimatie van de geometrische parameters, niet meer kunnen omgaan met deze complexere modellen. Sequentiele Monte Carlo methodes, een andere manier om recursieve Bayesiaanse estimatie kunnen dit wel, ten koste van een hogere rekencomplexiteit.

1.2 Een geïntegreerde aanpak van ACM

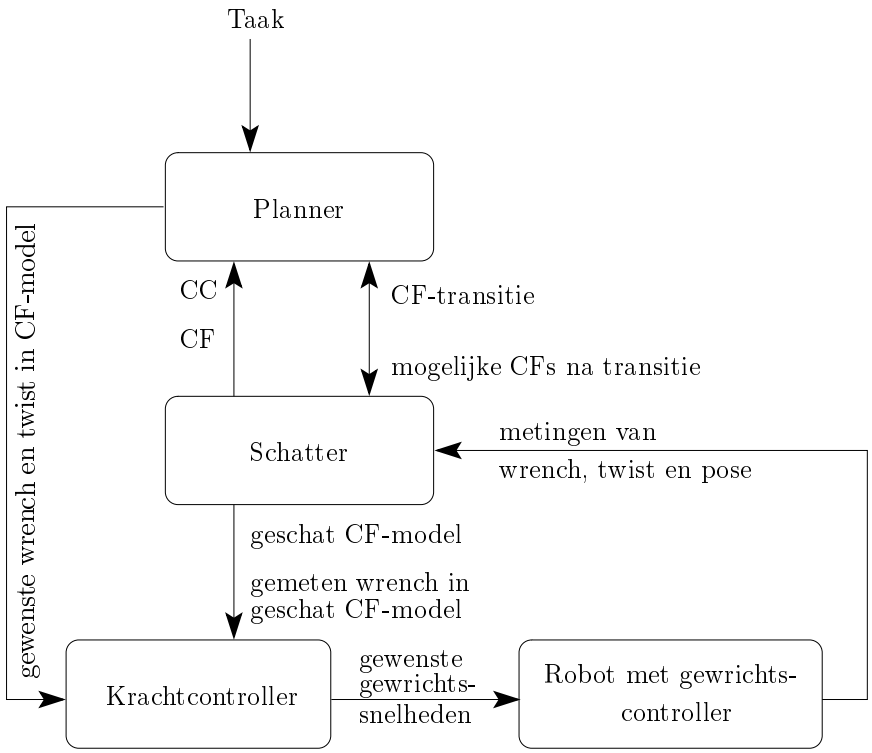
Figuur 2 toont het controle schema voor het ACM systeem dat ontwikkeld wordt aan ons departement. Het ACM systeem bestaat uit 3 componenten:

1. De controller berekent snelheidssetpoints die daarna naar de sturing van de robot gezonden worden. Deze controller gebruikt het hybride controle paradigma. Richtingen die niet beperkt zijn door contacten zijn snelheidsgecontroleerd, andere krachtgecontroleerd. De controller krijgt informatie van de planner en de schatter.

2. De schatter gebruikt kracht- en positiemetingen om hieruit informatie te verzamelen over de huidige (discrete) contactformatie en de waarde van de geometrische parameters.

3. De planner zet een hoog niveau taakplan om in setpoints voor de controller. Hierbij maakt hij gebruik van de informatie van een off-line gegenereerd taakplan en de resultaten van de schatter.

Deze thesis gaat voornamelijk over de schatter component, en werkt een strategie uit zodat de estimator kan omgaan met grote onzekerheden.



Figuur 2: Controleschema voor ACM systeem, bestaande uit 3 met elkaar communicerende componenten: controle, planning en schatters.

1.3 Bayesiaanse Waarschijnlijkheidleer

Deze thesis gebruikt Bayesiaanse waarschijnlijkheidsleer om onbekende variabelen te schatten. Bayesiaanse waarschijnlijkheidsleer biedt een consistent raamwerk om om te gaan met informatie (of onzekerheid). Het raamwerk verplicht om na te denken over alle veronderstellingen die gemaakt worden: de meeste algoritmes kunnen immers gekaderd worden binnen de Bayesiaanse statistiek mits het maken van specifieke veronderstellingen. Dit leidde in de praktijk reeds tot kruisbestuiving tussen verschillende onderzoeksdomeinen.

Alle problemen in deze thesis kunnen beschreven worden aan de hand van een a posteriori dichtheid van onbekende variabelen:

$$P(\boldsymbol{X}_{1:k} = \boldsymbol{x}_{1:k}, \boldsymbol{\Theta} = \boldsymbol{\theta} | \boldsymbol{Z}_{1:k} = \boldsymbol{z}_{1:k}), \tag{1}$$

warbij het subscript $_{1:k}$ duidt op een sequentie $_{1,2,\ldots,k}$. Deze thesis maakt expliciet het verschil tussen parameters ($\boldsymbol{\Theta}$) en toestanden ($\boldsymbol{X}_k$): De eerste zijn niet tijdsafhankelijk *in het beschouwde model*.[3] De reden voor deze opsplitsing is het feit dat sommige algoritmes enkel toepasbaar zijn voor ofwel parameters ofwel toestanden. Soms kan het nuttig zijn om $\boldsymbol{\Theta}$ en/of $\boldsymbol{X}_k$ verder op te splitsen in deelvariabelen $\boldsymbol{\Theta}^i$ en/of $\boldsymbol{X}_k^i$, omwille van conditionele onafhankelijkheidsrelaties tussen die delen. Bayesiaanse netwerken zijn een grafische voorstelling van een stochastisch model die deze relaties expliciteren, inzichten leveren in de structuur van deze modellen, en soms geautomatiseerde schattingsalgoritmes leveren voor deze modellen. Vectoren worden **vetjes** afgdrukt, en hoofdletters duiden op stochastische variabelen, terwijl kleine letters aangeven dat het om een concrete waarde van een bepaalde variabele gaat.[4]

Via de regel van Bayes kunnen we vergelijking (1) schrijven als

$$P(\boldsymbol{x}_{1:k}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta}) P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{P(\boldsymbol{z}_k | \boldsymbol{z}_{1:k-1})} P(\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta} | \boldsymbol{z}_{1:k-1}), \tag{2}$$

Deze recursie maakt gebruik van de Markov assumptie, die stelt dat volledige kennis over parameters en toestanden volstaat om te voorspellen wat de metingen zullen zijn. Bovenstaande vergelijking laat ons toe om *incrementeel* informatie uit nieuwe metingen toe te voegen aan reeds bestaande info over de onbekende variabelen. $P(\boldsymbol{z}_k | \boldsymbol{x}_k, \boldsymbol{\theta})$ wordt meestal omschreven als het *meetmodel*, en voorspelt hoe de metingen zullen zijn als we de toestanden en parameters uit ons systeem kennen. Het *systeemmodel* $P(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta})$ beschrijft de evolutie van toestand $k-1$ tot k, als we de toestand op $k-1$ en de waarde van de parameters exact kennen. Beide modellen kunnen extra (gekende) parameters bevatten. In het geval van het systeemmodel worden die

[3] Bemerk dat dit niet noodzakelijk overeenkomt met de fysische realiteit.

[4] Waar het onderscheid duidelijk is, worden hoofdletters weggelaten.

meestal *inputs* genoemd en genoteerd als $\boldsymbol{u}_{1:k} = \boldsymbol{u}_k \dots \boldsymbol{u}_k$. Bij meetmodellen spreekt met over *sensor parameters* $\boldsymbol{s}_{1:k} = \boldsymbol{s}_1 \dots \boldsymbol{s}_k$.

1.4 Filtering

Alhoewel vergelijking (2) toelaat om incrementeel informatie te verwerken, kan dit nog niet gebeuren in constante geheugenafmetingen, omdat alle toestanden vanop tijdstap 1 tot k worden bijgehouden. Dit kunnen we vermijden door op elke tijdstap te marginalizeren over de vorige toestand. Op die manier krijgen we algoritmes die online en in realtime metingen kunnen verwerken om informatie te leveren over de onbekende parameters en toestanden. Dit is immers wat we nodig hebben voor een volwaardig ACM systeem.

1.5 Bijdragen

Deze thesis levert bijdragen op het vlak van Bayesiaanse modelering, schatters voor het autonoom uitvoeren van krachtgecontroleerde robottaken en software voor recursief Bayesiaans schatten.

Bayesiaanse modelering. Deze thesis biedt een overzicht van estimatie-algoritmes, gebaseerd op de aard van de onbekende variabelen, en vergelijkt verschillende algoritmes en modellen met betrekking tot de veronderstellingen die ze maken. Dit overzicht is complementair aan bestaande overzichten gebaseerd op de representatie van de a posteriori dichtheid. Nieuwe modellen worden geintroduceerd, die kunnen gezien worden als een uitbreiding van Verborgen Markov Modellen of Jump Markov (lineaire) Systemen, waarbij een kruisafhankelijkheid bestaat tussen discrete modellen en continue parameters of toestanden. We tonen aan dat Sequentiële Monte Carlo methodes kunnen gebruikt worden om de onbekenden van die modellen Bayesiaans te schatten.

Schatters voor ACM. Deze thesis past de nieuwe modellen toe op het uitvoeren van krachtgecontroleerde robottaken onder grote onzekerheid, en bespreekt experimentele resultaten voor het concrete geval van een kubus-in-hoek assemblage. De hybride a posteriori dichtheid wordt recursief geschat via Particle Filters. Hiervoor worden expliciete meetvergelijkingen ontwikkeld uit bestaande impliciete modellen.

Software . Recursief Bayesiaans schatten van discrete, continue en hybride a posteriori kansdichtheden met verschillende algoritmes vereist een flexibel software raamwerk. BFL (Eng. Bayesian Filtering library, vrij vertaald: "Bayesiaanse filterbibliotheek") is een open bron software project that aan deze

eisen tegemoet komt en onafhankelijk is van één welbepaald toepassingsdomein. BFL werd gebruikt voor de estimatiealgoritmes in deze thesis, maar ook buiten PMA reeds succesvol toegepast in visie- en mobiele roboticatoepassingen. Deze thesis integreert BFL ook in de nieuw ontwikkelde robot software orocos. Dit vormt een belangrijke stap naar het volledig geïntegreerde ACM systeem uit figuur 2.

2 Monte Carlo methodes

Bayesiaanse waarschijnlijkheidsleer is niet verbonden aan één specifieke representatie. Eén mogelijke manier, om karakteristieken van de a posteriori dichtheid te berekenen is het gebruik van Monte Carlo methodes. Deze thesis gebruikt Monte Carlo methodes, omdat ze toegepast kunnen worden op de beschreven hybride modellen, in tegenstelling tot andere beschrijvingen zoals roostergebaseerde of analytische voorstellingen van de a posteriori dichtheid (beschreven in Sectie 3). Monte Carlo algoritmes dienen voornamelijk om numerieke integratie uit te voeren, en zijn in dat opzicht te vergelijken met meer gekende algoritmes zoals kwadratuurmethodes. Ze maken hiervoor gebruiken van zogenaamde (pseudo-) random number generatoren. In een Bayesiaanse context berekenen Monte Carlo methodes een schatting van de karakteristieken a posteriori dichtheid $p(\boldsymbol{x})$:

$$I = E_{p(\boldsymbol{x})}\left[h(\boldsymbol{x})\right] = \int h(\boldsymbol{x})p(\boldsymbol{x}), \tag{3}$$

voor continue kansdichtheden. Een vaak voorkomend probleem is de berekening van de verwachte waarde van de a posteriori dichtheid. In dat geval is $h(\boldsymbol{x}) = x$ en wordt (3):

$$E_{p(\boldsymbol{x})}\left[x\right] = \int \boldsymbol{x}p(\boldsymbol{x}). \tag{4}$$

Voor discrete en hybride dichtheden gelden gelijkaardige formules. Monte Carlo methodes benaderen deze karakteristieken door gebruik te maken van samples van de posterior:

$$I \approx \hat{I} = \sum_{i=1}^{N} h(\boldsymbol{x}^i) \qquad \boldsymbol{x}^i \sim p(\boldsymbol{x}), \tag{5}$$

waarbij de benadering convergeert als $N \to \infty$ en $\boldsymbol{x}^i \sim p(\boldsymbol{x})$ inhoudt dat $\boldsymbol{x}^i$ een i.i.d. sample is van $p(\boldsymbol{x})$.

Het grote voordeel van Monte Carlo methodes t.o.v. kwadratuur methodes is dat hun convergentie niet exponentieel vertraagt wanneer de dimensie van stijgt, alhoewel dit niet betekent dat ze noodzakelijk snel convergeren:

$$\sqrt{N}\left(\hat{I} - I\right) \to \mathcal{N}\left(0, \sigma^2\right), \tag{6}$$

waarbij $\sigma^2 = Var_{p(\boldsymbol{x})}[h(\boldsymbol{x})]$. Dus als de laatste term voldoende groot is hebben we nog steeds een convergentieratio van $\mathcal{O}\left(1/\sqrt{N}\right)$, maar niet noodzakelijk een snelle convergentie. Bovendien is het lang niet altijd triviaal om samples te genereren van $p(\boldsymbol{x})$.

2.1 Monte Carlo algoritmes

Verschillende algoritmes bestaan die toelaten om, al dan niet benaderend, samples te genereren van een bepaalde kansdichtheid $p(\boldsymbol{x})$. Grosso modo kan je Monte Carlo methodes indelen in iteratieve (beter gekend als MCMC, Markov ketting Monte Carlo) en niet-iteratieve methodes. Alhoewel de eerste duidelijk hun voordelen hebben, zijn we in het kader van deze thesis vooral geïnteresseerd in de niet-iteratieve methodes die we kunnen toepassen bij het online en recursief schatten. Hierbij is "importance sampling" (vrij vertaald, "samplen naar belangrijkheid") een belangrijk algoritme, voor het geval dat het moeilijk is om rechtstreeks samples te genereren van $p(\boldsymbol{x})$. Importance sampling maakt gebruik van een zogenaamde "proposal density" of voorstelkansdichtheid, waarvan wel makkelijk samples gegenereerd kunnen worden. Die samples krijgen dan een gewichtsfactor om de discrepantie tussen $p(\boldsymbol{x})$ en de proposal te compenseren. Importance sampling methodes leveren ook een alternatief om de convergentie van Monte Carlo methodes te versnellen door σ uit vergelijking (6) te verkleinen. De performantie van importance sampling methodes is sterk afhankelijk van hoe goed de proposal density $p(\boldsymbol{x})$ benadert in termen van Kullback-Leibler afstand en kan worden benaderd door het zogenaamd effectief aantal samples:

$$\frac{N}{1 + \sum_{i=1}^{N} \tilde{w}^i}, \tag{7}$$

waarbij N het aantal samples voorstelt en $\tilde{w}^i$ een genormaliseerde gewichtsfactor is die aangeeft hoe goed de proposal en $p(\boldsymbol{x})$ overeenkomen. Het effectief aantal samples geeft aan hoeveel samples van $p(\boldsymbol{x})$ nodig zouden zijn om eenzelfde precisie te bekomen als voor het schatten van integraal (3) met N samples van de proposal. Een groot voordeel van importance sampling methodes is ook dat het genereren van een gegeven aantal samples steeds in een constante tijdsinterval kan gebeuren. Dit is niet het geval voor andere methodes zoals rejection sampling of MCMC methodes. Daarom ligt het importance sampling algoritme aan de basis van sequentiële Monte Carlo methodes, meestal omschreven als Particle Filters.

2.2 Sequentiële Monte Carlo methodes

Aangezien ACM nood heeft aan online, recursieve schatters, is het belangrijk om Monte Carlo methodes toe te passen op de recursieve toepassing van de stelling van Bayes (vergelijking. Sequentiële Monte Carlo methodes (beter bekend onder de naam Particle Filters) gebruiken een recursieve versie importance sampling methodes om vergelijking (2) op te lossen. Door ook de proposal dichtheid op een recursieve manier op te stellen, bekomen we hoe we de gewichten van de samples moeten updaten bij elke tijdstap:

$$w_k(\boldsymbol{x}_k, \boldsymbol{\theta}) \sim \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{\theta})P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta})}{q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k)} w_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{\theta}), \tag{8}$$

waarbij $q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k)$ de incrementele proposal dichtheid voorstelt.

Keuze van de proposal dichtheid. Net zoals bij standaard importance sampling, speelt de keuze van de proposal dichtheid ook bij sequentiële Monte Carlo methodes een belangrijke rol in de convergentie van de algorithmes. Doucet, Godsill, and Andrieu (2000) toont aan dat de optimale (i.e. de proposal die de a posteriori Monte Carlo covariantie minimaliseert) proposal dichtheid voor een Markoviaans systeem gelijk is aan

$$\pi(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_{1:k}) = P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{\theta}, \boldsymbol{z}_k). \tag{9}$$

Bemerk dat de optimale proposal dichtheid rekening houdt met de nieuwe meting $\boldsymbol{z}_k$. Jammer genoeg is het in de meeste gevallen heel moeilijk om samples te genereren van π. Daarom wordt er meestal een benadering gemaakt van de optimale proposal dichtheid. Verschillende versies van Sequentiële Monte Carlo methodes verschillen vaak alleen door een andere keuze van de proposal dichtheid.

Verarming. In de praktijk zullen, naarmate het aantal metingen stijgt, de gewichten heel onevenredig verdeeld worden, doordat proposal en a posteriori dichtheid uit elkaar groeien. Eén of enkele samples krijgen een groot gewicht, terwijl dat van de andere heel klein wordt. Dit fenomeen noemt men verarming (Eng. *impoverishment*) en resulteert in een grote Monte Carlo covariantie voor de schattingen. Om dit te vermijden bestaan een aantal technieken. Hersamplen van de discrete sample set is hier de meest voor de hand liggende optie. Dit kan op een aantal manieren gebeuren: zowel het gebruikte algoritme als het tijdstip van resamplen resulteren in andere versies van het Particle Filter algoritme. Een andere manier om verarming tegen te gaan, die kan gecombineerd worden met hersamplen, is het toepassen van een MCMC stap. In de meeste gevallen leidt dat echter tot een algoritme dat niet meer in een constant tijdsinterval kan uitgevoerd worden.

Convergentie. Gezien het feit dat sequentiële Monte Carlo methodes slechts recent (her)ontdekt worden, zijn de convergentieresultaten nog beperkt. Crisan and Doucet (2002) bewijzen dat de MSE naar nul convergeert met een ratio $\mathcal{O}(1/N)$:

$$E[(\hat{I}_k - I_k)^2] \leq \frac{c_k}{N}, \tag{10}$$

waarbij I_k de gewenste karakteristiek van de posterior voorstelt:

$$I_k = \int P(\boldsymbol{x}_k, \boldsymbol{\theta} | \boldsymbol{z}_{1:k}) \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{\theta}) \, d\boldsymbol{x}_k d\boldsymbol{\theta}, \tag{11}$$

$\hat{I}_k$ de Monte Carlo schatting, en c_k een tijdsvariabele constante. Jammer genoeg geldt dit resultaat enkel voor begrensde functies $\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{\theta})$, wat betekent dat convergentie voor bv. de verwachte waarde nog niet bewezen is. Bovendien is c_k tijdsvariabel. Vooral bij statische systemen en een groot aantal metingen kan dit problemen opleveren. Ondanks dit alles toont deze thesis en ander onderzoek aan dat Particle Filters wel degelijk bruikbaar zijn voor problemen waar andere filteralgoritmes falen.

3 Classificatie van Bayesiaanse algoritmes

Deze sectie geeft een overzicht van mogelijke implementaties van recursief Bayesiaans schatten, gebaseerd op de aard van de parameter- en/of toestandsvector. De nadruk van deze sectie ligt op

- Toepassingen in de robotica in het algemeen en ACM in het bijzonder;
- state-of-the-art Monte Carlo technieken.

Het doel van deze sectie is een soort referentietabel te bieden, die helpt om te kiezen welke algoritmes van toepassing zijn, eens men een bepaald stochastisch model gekozen heeft. Het overzicht verheldert ook de relatie tussen modelkeuze en rekenkundige complexiteit van (online) implementaties van de besproken algoritmies. Dit overzicht is complementair aan bestaande overzichten, wiens classificatie gebaseerd is op de voorstelling van de a posteriori dichtheid. Grosso modo kunnen we op die manier filtering algoritmes indelen als volgt:

1. **Analytische algoritmes** beschrijven de a posteriori dichtheid door een analytische functie. Het best gekende voorbeeld is de Kalman Filter, die de a posteriori dichtheid voorstelt als een unimodale Gaussiaanse dichtheid. Alle algoritmes die gebaseerd zijn op het principe van conjugatie. Per definitie kunnen analytische algoritmes niet opgaan met discrete onbekenden. Hybride variabelen worden meestal geschat door een aantal filters in parallel te gebruiken.

2. **Roostergebaseerde algoritmes** stellen de posterior voor door een uniform rooster. De granulariteit van dit rooster is soms tijdsafhankelijk, wat meestal omschreven wordt als adaptieve roostergebaseerde algoritmes. Deze aanpak is niet gelimiteerd tot variabelen van een bepaalde aard, maar scaleert heel slecht naar mate de dimensie van de parameter- en/of toestandsvector stijgt.

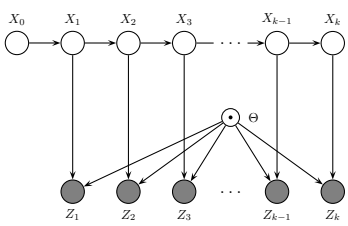
3. **Monte Carlo algoritmes** (beschreven in de vorige sectie) beschrijven de posterior als een verzameling van *random* samples. Net als roostergebaseerde kunnen ze elk type variabelen aan, en zijn ze ook niet beperkt tot systemen die voldoen aan het principe van conjugatie. Bovendien scaleert de aanpak beter dan roostergebaseerde methodes naarmate de dimensie van de parameter- en/of toestandsvector stijgt. Ze zijn echter wel rekenkundig complexer en omwille van de discretisatie moet men soms oppassen voor numerieke problemen.

Bovenstaande opsomming is beperkt tot volledige Bayesiaanse schatters: Alle algoritmes gebruiken a priori informatie en leveren een (benaderende) beschrijving van de posterior. Voor een gegeven problemen, gebeurt het vaakt dat volledig Bayesiaanse schatters te rekenintensief zijn om nuttig gebruikt te worden. Om dit te verhelpen bestaan een aantal benaderende methodes, zoals Maximum Likelihood (ML of Maximum A Posteriori (MAP) methodes. Een typisch voorbeeld is het EM algoritme, een off-line optimisatie algoritme dat gebruikte wordt om de posterior over onbekende parameters te vervangen door een ML schatter, waarna deze ML schatting gebruikt kan worden in een volledige Bayesiaanse schatters tijdens online estimatie.

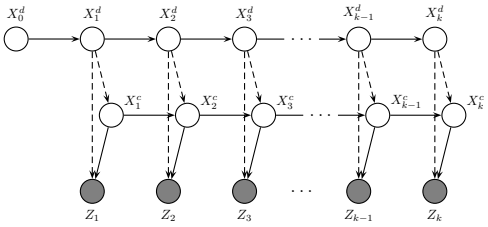
Tabel 5.1 geeft een (Engelstalig[5]) tweedimensionaal overzicht van de algoritmes gebaseerd op de aard van de parameter- en/of toestandsvector. Samen met bovenstaande classificatie op basis van de representatie van de posterior, resulteert dit in een driedimensionale tabel. De tabel bevat 15 ($4 \times 4 - 1$—in het geval dat zowel $\boldsymbol{x}$ als $\boldsymbol{\theta}$ gekend zijn—) gevallen van mogelijke $\boldsymbol{x}/\boldsymbol{\theta}$ combinaties. De eerste kolom bevat alle gevallen waar geen toestandsvariabelen geschat moeten worden. Het gaat hier om pure parameterschatting. Analoog beschrijft de eerste rij pure toestandsestimatieproblemen.

Een volledige bespreking van de tabel valt buiten dit overzicht. Door de rigoureuze opsplitsing op basis en de voorstelling aan de hand van Dynamische Bayesiaanse Netwerken (DBN) kwamen wel 2 DBN topologieën aan het ligt, die voor zover ik weet, nooit eerder gebruikt zijn. Het gaat om een uitbreiding van Jump Markov Systemen en Verborgen Markov Modellen met onbekende parameters die toelaten dat de evolutie van de discrete toestandsvector ook beïnvloed wordt door de waarde van de continue toestand of parameter. Figuur 3 illustreert dit aan de hand van een DBN voorstelling.

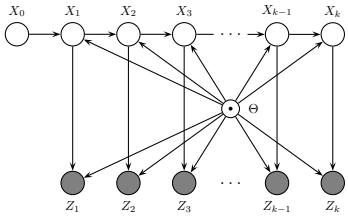
[5]de meeste algoritmes hebben niet direct een Nederlands equivalent



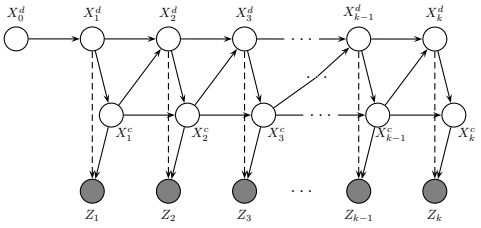
(a) Verborgen Markov model met onbekende parameters. In dit model kan de evolutie van de discrete toestanden niet afhangen van de waarde van de onbekende parameters.



(b) Jump Markov Systeem. De evolutie van de discrete toestanden wordt niet beïnvloed door de waarde van de onbekende toestanden.



(c) Verborgen Markov Model met onbekende continue parameters en kruisafhankelijkheid tussen parameters en toestanden.



(d) Uitbreiding op Jump Markov Systemem. Dit model laat afhankelijkheid toe tussen het discrete en continue deel van de toestandsvector.

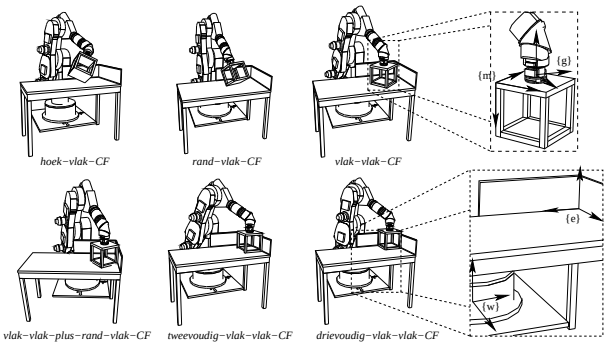
Figuur 3: Expliciete hybride Bayesiaanse modellen, die toelaten dat de evolutie van de discrete toestand beïnvloed wordt door de waarde van de continue parameter- of toestandsvector.

4 Toepassing: hybride model-parameter estimatie bij assemblage

Deze sectie past een specifiek geval uit de vorige sectie toe op één bepaalde ACM taak: het gelijktijdig schatten van geometrische parameters en Contactformaties tijdens de assemblage van een kubus-in-hoek-assemblage.

4.1 Probleemomschrijving

Figuur 1 toont de opstelling voor dit experiment, samen met een verklaring van de gebruikte termen, en figuur 4 toont hoe dit experiment opgebouwd is uit een sequentie van een aantal *contactformaties*, zoals het punt-vlak contact waarbij een punt van de kubus in contact is met een vlak van de hoek. De positie en oriëntatie van de kubus t.o.v. de eindeffector van de robot en



FIGUUR 4: Autonome cubus-in-hoek assemblage door een seriele robot: verschillende *contactformaties* (CFs).

de positie en oriëntatie van de hoek t.o.v. een vast wereldassenstelsen zijn ongekend. Deze onbekenden worden vaak omschreven als de *geometrische parameters* en zijn continu van aard. De krachtsensor en de encoders leveren metingen die gebruikt worden om de waarden van deze parameters te schatten. Elke discrete contactformatie levert een verschillend model op dat geometrische parameters linkt aan de metingen. Vroeger onderzoek veronderstelt dat, van zodra een consistentietest aangeeft dat het huidige model niet meer geldig is, de volgende contactformatie gekend is. Dit is alleen geldig wanneer het experiment uitgevoerd wordt met kleine onzekerheden op de geometrische parameters. De methode beschreven in deze sectie laat assemblage toe onder grotere onzekerheden.

4.2 Hybride aanpak

Om te kunnen omgaan met grotere onzekerheden, maakt dit werk gebruik van een hybride (gedeeltelijk discreet, gedeeltelijk continu) a posteriori dichtheid over continue geometrische parameters en discrete contactformaties

$$P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_k = j \mid \boldsymbol{Z}_{1:k} = \boldsymbol{z}_{1:k}\right). \tag{12}$$

Wanneer we (12) recursief ontwikkelen door gebruik te maken van de stelling van Bayes, bekomen we voor de meetupdate

$$P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k}\right) = \frac{P\left(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k = j\right) P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right)}{P\left(\boldsymbol{z}_k \mid \boldsymbol{z}_{1:k-1}\right)}. \tag{13}$$

Uitgaande van $P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right)$, kunnen we de a posteriori dichtheid op tijdstap k berekenen door de informatie uit onze meting $\boldsymbol{z}_{1:k}$ te verwerken: $P\left(\boldsymbol{z}_k \mid \boldsymbol{\theta}, CF_k = j\right)$ beschrijft hoe waarschijnlijk een bepaalde meting is, gegeven een bepaalde waarde van de geometrische parameters en een bepaalde contactformatie. De *systeemupdate* beschrijft hoe we $P\left(\boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right)$ kunnen berekenen uitgaande van de a posteriori dichtheid op tijdstap $k-1$:

$$\begin{aligned} &P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_k = j \mid \boldsymbol{z}_{1:k-1}\right) \\ &\quad = \sum_i \left[P(CF_k = j \mid CF_{k-1} = i, \boldsymbol{\Theta} = \boldsymbol{\theta}) P\left(\boldsymbol{\Theta} = \boldsymbol{\theta}, CF_{k-1} = i \mid \boldsymbol{z}_{1:k-1}\right)\right]. \end{aligned} \tag{14}$$

De systeemupdate gebruikt een expliciet hybride model $P(CF_k = j \mid CF_{k-1} = i, \boldsymbol{\Theta} = \boldsymbol{\theta})$ dat CF transities beschrijft uitgaande van de waarde van de geometrische parameters en de huidige contactformatie. Inderdaad, wanneer we gebruik maken van een kinematisch model van de robot en rekening houden met de snelheden die we naar de motoren sturen, kunnen we contactformatietransities voorspellen. Dit model wordt weergegeven door figuur 6.5(a) op pagina 112. De huidige implementatie maakt echter nog gebruik van een vereenvoudigd model, dat CF transities modelleert aan de hand van een verborgen Markov Model

$$P(CF_k = j \mid CF_{k-1} = i, \boldsymbol{\Theta} = \boldsymbol{\theta}) \approx P(CF_k = j \mid CF_{k-1} = i). \tag{15}$$

Deze vereenvoudiging komt niet goed overeen met de werkelijkheid, omdat de kans om een gegeven tijdsduur in een contactformatie te blijven exponentieel afneemt. Echter, uit de experimentele resultaten blijkt dat voor relatief kleine onzekerheden genoeg informatie aanwezig in de metingen aanwezig is om dit slechte systeemmodel te compenseren. Echter, wanneer zal de implementatie van het hybride model zorgen dat minder rekenkracht nodig is om goede schattingen te bekomen. Figuur 6.5(b) geeft een DBN voorstelling van dit vereenvoudigd model, dat neerkomt op een statische versie van Jump Markov systemen.

4.3 Algoritmes

De hybride posterior dichtheid legt niet vast welke filter gebruikt moet worden om de metingen online te verwerken. De Kalman Filters die vroeger gebruikt werden kunnen voor deze modellen niet gebruikt worden. Inderdaad, ondanks het feit dat voor Jump Markov systemen een aantal filters ontwikkeld zijn die een aantal Kalman Filters in parallel gebruiken en op elk tijdstip een reductie van het aantal modes doorvoeren,[6] zijn deze filters niet in staat om dat te doen voor sterk niet-lineaire systemen, tenzij er gewerkt wordt met kleine onzekerheden. Om met die grote onzekerheden te kunnen omgaan, ontwikkelde Lefebvre, Gadeyne, Bruyninckx, and De Schutter (2003) de niet-minimale toestands Kalman Filter, een variant op de Kalman Filter. Door de geometrische parameters te transformeren naar een hoger dimensionale ruimte ontstaat daar een lineair systeem. Dit vermijdt een opeenstapeling van opeenvolgende fouten door de linearisatie. Er is echter een verschillende transformatie nodig voor elke contactformatie, wat het onmogelijk maakt om schattingen van de ene naar de andere contactformatie te propageren bij contactformatietransities onder grote onzekerheid. Meer nog, geen enkele van de Kalman Filters kan omgaan met het volledig hybride model $P\left(CF_k = j \mid \boldsymbol{\Theta} = \boldsymbol{\theta}', CF_{k-1} = i\right)$. Alhoewel de huidige resultaten deze hybride transitiedichtheid nog niet gebruiken, zou dit het onmogelijk maken om dit accurate contactformatietransitiemodel te gebruiken in toekomstig onderzoek.

Sequentiële Monte Carlo kunnen deze hybride modellen wel aan, alhoewel dat een verhoogde rekenkracht met zich meebrengt, en er zich een aantal maatregelen opdringen om numerieke problemen te vermijden, des te meer ook parameters moeten geschat worden. Bovendien moeten hiertoe expliciete meetmodellen ontwikkeld worden, omdat alleen Kalman Filters de impliciete modellen die tot nu toe gebruikt werden kunnen verwerken. Deze expliciete modellen worden bekomen een transformatie van de metingen naar een andere ruimte voor de snelheids- en krachtmetingen, waarbij de nieuwe meting de energie voorstelt, en in het ideale geval 0 moet zijn in alle richtingen, gezien het reciprociteitsbeginsel. Voor de posemetingen, wordt elke contactformatie beschreven als een geheel van al dan niet gemaakte punt vlak contacten. Als een punt-vlak contact gemaakt is, wordt verondersteld dat de afstand punt-vlak normaal verdeeld is rond nul, met een kleine onzekerheid. Als een punt-vlak contact niet gemaakt is, wordt een veel grotere onzekerheid genomen (wat in de praktijk neerkomt op een benadering van een uniforme dichtheid.

[6] Inderdaad, uit vergelijking 14 blijkt dat het aantal modes exponentieel toeneemt met de tijd.

4.4 Experimentele resultaten

De experimenten tonen aan dat er goede resultaten bekomen worden voor schattingen met matige onzekerheid, zowel voor de schatting van de contactformaties als de schatting van de geometrische parameters. Om dit uit te breiden naar nog grotere onzekerheden en toch nog de metingen online te kunnen verwerken, zijn 2 zaken noodzakelijk zijn:

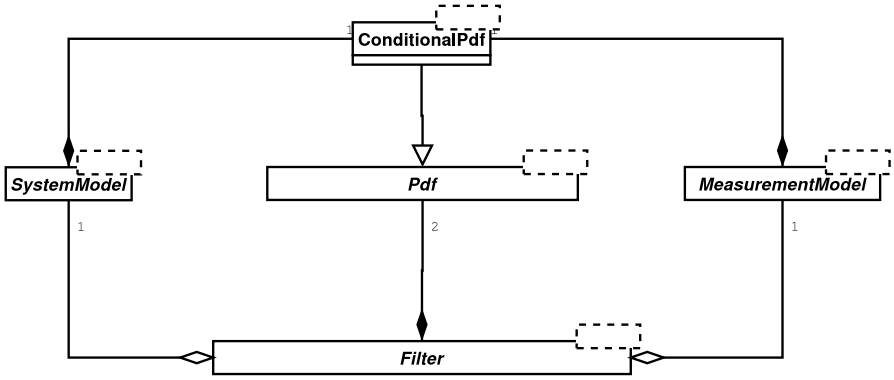
- Het gebruik van aangepaste Particle Filters, die gebruik maken van meetinformatie tijdens hun voorstel-stap.
- Het gebruik van het accurate hybride transitiemodel, dat vermijdt dat onnodig deeltjes gesampled worden in contactformaties die wel mogelijk zijn volgens de systeemgrafe, maar niet wanneer rekening gehouden wordt met de huidige schatting van geometrische parameters en contactformaties.

Uit de resultaten blijkt ook dat actief waarnemen onmisbaar is om accurate schattingen te bekomen. Immers, de meetdata die gebruikt werden voor dit experiment waren niet informatief, met een vrij grote onzekerheid tot gevolg in specifieke richtingen die niet genoeg geëxciteerd werden.

5 BFL, een bibliotheek voor recursieve Bayesiaanse algoritmes

De vorige secties illustreren dat er heel wat onderzoek plaatsvindt naar Bayesiaanse toestands- en parameterschatting. Bijgevolg is er ook heel wat software hiervoor beschikbaar. De meeste Bayesiaanse bibliotheken zijn echter geprogrammeerd met één bepaalde toepassing of algoritme in gedachten, wat het moeilijk maakt om ze te herbruiken. Gedurende deze thesis ontwikkelde ik *BFL* (The "Bayesian Filtering Library", letterlijk vertaald de "bibliotheek voor recursief Bayesiaans schatten") (Gadeyne 2001b), een C^{++} bibliotheek die ontworpen werd met volgende vereisten in gedachten:

Bayesiaans De bibliotheek moet het mogelijk maken om alle recursieve volledig Bayesiaanse schatters beschreven in sectie 3 te implementeren. Dit houdt in dat de bibliotheek geen beperkingen mag opleggen met betrekking to (i) de aard van de te schatten variabelen (discrete, continu of hybride), (ii) noch op de voorstelling van de posterior dichtheid (analytisch, Monte Carlo voorstelling, ...). Dit moet toelaten om de verschillende algoritmes te gebruiken met een maximaal hergebruik van bestaande code en de performantie van verschillende algoritmes makkelijker met elkaar te vergelijken.



FIGUUR 5: Ontwerp van BFL, voorgesteld in een UML diagramma.

Open Een veelvoorkomend probleem in de robotica is het gebrek aan gestandaardiseerde data sets zoals die gebruikt worden in andere takken van de wetenschap. Hetzelfde geldt op niveau van algoritmes: Een gemeenschappelijk software raamwerk voor alles recursieve Bayesiaanse schatters laat toe om makkelijker verschillende (implementaties van) algoritmes met elkaar te vergelijken. *Open bron software* (Open Source Initiative) biedt enorme perspectieven hiervoor. Bovendien ligt open bron software aan de basis van "reproduceerbaar onderzoek" (Buckheit and Donoho 1995), wat toelaat om onderzoeksresultaten snel en makkelijk te reproduceren.

Onafhankelijk Op dit moment is er geen *standaard* bibliotheek voor numerieke of stochastische doeleinden beschikbaar. Omdat een bibliotheek met schattingsalgoritmes meestal maar een deel van de software in een systeem uitmaakt, en bestaande onderdelen reeds een bepaalde numerieke of stochastische bibliotheek gebruiken, moet het mogelijk zijn die te herbruiken voor het estimatiegedeelte.

Onafhankelijk houdt ook in that de bibliotheek niet gekoppeld mag zijn aan één specifieke toepassing en dus zijn interface en implementatie los moeten staan van specifieke sensoren, veronderstellingen en algoritmes.

Om vlot te integreren in onze bestaande robot controlesoftware, werd C++ gekozen als programmeertaal voor BFL.

BFLs ontwerp voldoet aan bovengestelde eisen. Figuur 5 toont de basisklassen van BFL en hoe ze met elkaar in relatie staan. Elke filter implementeert de `Filter<T1,T2>` interface, waarbij `T1` en `T2` respectievelijk de aard van de gecombineerde toestand-parametervector en de meting vastleggen, die zowel discreet, continu als hybride kunnen zijn. De a priori en a posteriori

kansdichtheden zijn implementaties van de `Pdf<T>` interface, waarbij `<T>` het template type van de gecombineerde toestand-parametervector voorstelt. Er wordt geen onderscheid gemaakt tussen parameters en toestanden: dat is aan de onderzoeker. Vermits `Pdf` enkel een interface klasse is, legt ze niet vast hoe de informatie uit de kansdichtheid wordt bijgehouden.

Systeem- en meetmodel gebruiken de `ConditionalPdf<T1,T2>` klasse, die alle kansdichtheden van het type $P(A|B)$ voorstelt waarbij A van type `T1` en B van type `T2` is.

Door de getemplatiseerde aanpak voldoet BFL aan de eerste vereiste. Om onafhankelijk te zijn van één bepaalde numerieke en/of stochastische bibliotheek, laat BFL toe om een abstractielaag te schrijven rond bestaande bibliotheken.

BFL is beschikbaar via het internet onder een open bron licentie. De voornaamste reden hiervoor is dat ik achter de gedachte van reproduceerbaar onderzoek sta. Bovendien verplicht open bron code to propere en goed gedocumenteerde code en worden programmeerfouten sneller ontdekt.

BFL wordt niet alleen aan het departement werktuigkunde gebruikt voor het autonoom uitvoeren van robottaken in contact met de omgeving, maar over de ganse wereld en voor een breed spectrum aan toepassingen zoals robuuste visie, mobiele robotica, en. Meer informatie over deze toepassingen is beschikbaar via (Gadeyne 2001b). BFL is gedurende deze thesis ook geintegreerd in het nieuwe orocos raamwerk voor open robot controle.

6 Conclusies

Het doel van deze thesis was na te gaan of Sequentiële Monte Carlo methodes een toegevoegde waarde vormden in het ACM onderzoek op het departement werktuigkunde. Om ACM taken uit te kunnen voeren in onzekere of onbekende omgevingen, moeten parameters en of toestanden geschat worden. Dit werk focust op—maar is niet beperkt tot—het schatten van continue *geometrische parameters* en discrete, tijdsvariabele *contactformaties*.

Vroeger onderzoek ontwikkelde Bayesiaanse modellen die positie- en krachtmetingen van de robot relateerden aan de onbekende geometrische parameters en contactformaties. De niet-minimale-toestand-Kalman-Filter levert nauwkeurige schattingen van de geometrische parameters, als deze observeerbaar zijn en de posterior geconvergeerd is naar een unimodale Gaussiaanse dichtheid, zelfs voor grote onzekerheden op de geometrische parameters. Echter, bij grote initiële onzekerheden is er ook ambiguiteit over welke contactformatie (of anders gezegd, welk discreet meetmodel) de metingen veroorzaakt. De consistentietesten uit vroeger onderzoek zijn hier niet meer toepasbaar. Het detecteren van die ambiguiteit is ook belangrijk om de meest geschikte *actief waarnemen* strategie te bepalen.

6.1 Bijdragen van dit werk

Misschien is de belangrijkste bijdrage van dit werk wel zijn loyaliteit aan Bayesiaanse waarschijnlijkheid in al zijn facetten. Letterlijk gezien vanuit een algoritmisch standpunt, maar meer belangrijk vanuit een practisch oogpunt, door alle assumpties expliciet te maken, zowel op vlak van modellen, algoritmes en software. Dit alles wil bijdragen aan het idee van reproduceerbaar onderzoek. De volgende secties bespreken kort de belangrijkste meer voor de hand liggende bijdragen van dit onderzoek.

Overzicht van Bayesiaans schatten op basis van de aard van de random variables

Dit werk geeft een interdisciplinair overzicht van algoritmes voor het schatten van de a posteriori dichtheid $P(\boldsymbol{x}, \boldsymbol{\theta})$. In tegenstelling met, en complementair aan, bestaande overzichten die gebaseerd zijn op de voorstelling van algoritmes op basis van hoe de de a posteriori dichtheid voorstellen, is de classificatie van dit overzicht gebaseerd op de aard van de ongekende variabelen (discreet, continu of hybride, i.e. gedeeltelijk discreet, gedeeltelijk continu). Pure toestands- of parameterschatting zijn belangrijk bijzondere gevallen. De focus van het overzicht ligt op recursieve Bayesiaanse schatters, maar links met off-line trainingsalgoritmes en semi-deterministische benaderingen worden gemaakt. Op die manier is het mogelijk om een algoritme te kiezen uit een overzichtelijke tabel, eens gekozen is voor een bepaald model. Het overzicht verduidelijkt ook de relatie tussen modelkeuze en de complexiteit van de bijhorende algoritmes.

De classificatie op basis van de aard van de ongekende variabelen leidt ook tot het ontdekken van 2 nieuwe Bayesiaans modellen, die een uitbreiding zijn van resp. Verborgen Markov Modellen met ongekende parameters en Jump Markov Systemen. Sequentiële Monte Carlo methodes worden gebruikt om de posterior te schatten in dergelijke systemen.

ACM

Deze thesis ontwikkelt een expliciet hybride Bayesiaans model om contactformatietransities te beschrijven. Vroeger onderzoek aan het departement gebruikte consistentietesten hiervoor of maakte gebruik van verborgen Markov Modellen. Consistentietesten modelleren geen contactformatietransities, ze detecteren via een drempel hoe waarschijnlijk het is dat de huidige metingen afkomstig zijn van een bepaald model. Daardoor kunnen ze niet omgaan met ambiguïteiten in contactformaties en zijn ze alleen toepasbaar bij kleine initiële onzekerheden. Verborgen Markov Modellen zijn slechts een *heel* ruwe benadering van contactformatietransities. Het hybride transitiemodel uit de-

ze thesis volgt natuurlijk uit de volledig Bayesiaanse recursie van de hybride posterior dichtheid over geometrische parameters en contactformaties.

Wanneer de parametervector vervangen wordt door een toestandsvector, is dit hybride model ook toepasbaar op hybride toestandsestimatie. Dit is een uitbreiding op Jump Markov Modellen, die vaak gebruikt worden in de literatuur voor hybride toestandsestimatieproblemen. Deze thesis beschrijft hoe dit model zou kunnen toegepast worden op gelijktijdige toestandsestimatie en foutdiagnose.

The niet-minimale-toestand-Kalman filter, die goede resultaten oplevert voor pure parameterestimatieproblemen, ondanks een lage complexiteit, kan niet gebruikt worden in deze hybride modellen. Sequentiële Monte Carlo methodes zijn wel toepasbaar, ten koste van extra rekenkracht. Dit werk ontwikkelt expliciete meetmodellen voor de Sequentiële Monte Carlo methodes op basis van bestaande impliciete vergelijkingen.

Software

De classificatie van Bayesiaanse algoritmes voor recursief Bayesiaans schatten legt de nood bloot voor een flexibel software raamwerk: Een Bayesiaanse bibliotheek mag geen beperkingen opleggen met betrekking tot (i) de aard van de aard van de onbekenden (discrete, continue of hybride parameters en/of toestanden) noch (ii) de voorstelling van de a posteriori dichtheid (analytisch, Monte Carlo, roostergebaseerd, ...).

BFL voldoet aan deze vereisten. Zijn volledige getemplatiseerd ontwerp omwat all mogelijke gevallen beschreven in sectie 3 en de abstracte `Pdf` interface laat toe om het concept van een kansdichtheid te scheiden van zijn implementatie. Dit resulteert in een maximum aan herbruikbare code wanneer verschillende filter algoritmes gebruikt worden voor één problem. BFL gebruikt het concept van abstractielagen om onafhankelijk te zijn van een bepaalde numerieke en stochastische bibliotheek, wat de integratie in bestaande software projecten makkelijker maakt. De bibliotheek is ook een open bron software project, hiermee aanleunend bij het concept van reproduceerbaar onderzoek.

Dit werkt toont hoe BFL kan geïntegreerd worden in de orocos controle software en schetst een aantal applicaties die BFL gebruiken en niet beperkt blijven tot ACM.

6.2 Beperkingen en toekomstig onderzoek

Ondanks zijn bijdragen, vormt dit werk maar een kleine stap naar het volledige geïntegreerde ACM systeem uit figuur 2.

Grotere onzekerheden

Ondanks het feit dat, voor de onzekerheden die in dit werk gebruikt werden, de software metingen kon verwerken a rato van 0.5Hz met 5000 samples op een 1.1 GHz pc, verwacht ik dat hier nog aanzienlijke verbeteringen mogelijk zijn. Dit is nodig om real-time te kunnen omgaan met grotere onzekerheden, ermee rekening houdend dat de estimator slechts één component is van het ACM systeem. In het bijzonder zijn volgende twee zaken belangrijk:

- De *implementatie* van het hybride transitiemodel om contactformaties te voorspellen. In dit model moeten de snelheidssetpoints die naar de robot gestuurd worden gebruikt worden.
- Betere proposal dichtheden voor de Sequentiële Monte Carlo methods zullen nodig zijn om divergentie van het algoritme te vermijden bij grotere onzekerheden.

Actief waarnemen

Alhoewel deze thesis wel wat hints bezorgt over het gebruiken van covariantie en entropie voor actief-waarnemen-strategieën, gaat het hier niet dieper op in. Het lijkt alsof POMDP algoritmes niet toepasbaar zijn omwille van hun enorme complexiteit. Daarom zullen meer ad hoc strategieën nodig zijn die rekening houden met een set van karakteristieken van de a posteriori dichtheid en deze informatie in de online (her)planner verwerken.

Integratie van de ACM componenten en online experimentele validatie

Nu de robotcontrolesoftware en de estimatiesoftware een vrij "rijpe" toestand bereikt hebben, en de integratie van de gegevens van de off-line planner omgezet kunnen worden naar bruikbare setpoints voor de controller, lijkt het mogelijk om binnen een redelijke een experimentele validatie te doen van het kubus-in-hoek assemblage experiment. Hiertoe moet de planner nog aangepast worden om de gegevens van de schatter te gebruiken en een actief waarnemen strategie bedacht worden die herplant aan de hand van de data van de estimator. Dit zal toelaten om na te gaan hoe robuust alles is met betrekking tot de keuze van de parameters en zal toekomstig *porteren* naar gelijkaardige applicaties vergemakkelijken.