

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT WERKTUIGKUNDE
AFDELING PRODUCTIETECHNIEKEN,
MACHINEBOUW EN AUTOMATISERING
Celestijnenlaan 300B, B-3001 Heverlee (Leuven), België

STRUCTURED LIGHT ADAPTED TO CONTROL A ROBOT ARM

Promotor :
Prof. dr. ir. H. Bruyninckx

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Kasper Claes

2008D04 Mei 2008



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT WERKTUIGKUNDE
AFDELING PRODUCTIETECHNIEKEN,
MACHINEBOUW EN AUTOMATISERING
Celestijnenlaan 300B, B-3001 Heverlee (Leuven), België

STRUCTURED LIGHT ADAPTED TO CONTROL A ROBOT ARM

Jury :
Prof. dr. ir. Y. Willems, voorzitter
Prof. dr. ir. H. Bruyninckx, promotor
Prof. dr. ir. J. De Schutter
Prof. dr. ir. L. Van Gool
Prof. dr. ir. D. Vandermeulen
Prof. dr. ir. J. Baeten
Prof. dr. ir. P. Jonker
Technische Universiteit Eindhoven, Nederland

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Kasper Claes

U.D.C. 681.3*I29
Wet. Depot: D/2008/7515/10
ISBN 978-90-5682-901-8

Mei 2008

© 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/2008/7515/10
ISBN 978-90-5682-901-8
UDC 681.3*I29

Voorwoord

"Is dat dan een voltijdse job, die reisbegeleiding?"
"Nee, da's maar een hobby."
"Wat voor werk doe je dan?"
"Ach, iets heel anders: de interpretatie van videobeelden in robotica."
"Ah, domotica, daar heb'k al's van gehoord . . . "
"Nee, robotica*, met een robotarm." "Een robotarm?"*

"Ga je na dat doctoraat dan op zoek naar een job?"
"Voor mij is dat doctoraat altijd een job geweest, en een leuke ook. . . "

Toen ik in het voorjaar van 2004 op zoek was naar een interessante job had ik geen idee wat een doctoraat zou kunnen inhouden. Het was een sprong in het onbekende waar ik geen moment spijt van heb gehad. Jaren die m'n wereld letterlijk en figuurlijk verruimden: de universitaire wereld, de open-source-wereld, computervisie, onderzoek. . . een boel boeiend materiaal waar ik tevoren nog geen kaas van gegeten had. Bedankt Herman en Joris, om die deuren voor me open te zetten.

Wie ik zeker ook wil bedanken, zijn m'n directe (ex-)collega's, met elk hun heel eigen persoonlijkheid, en toch zoveel om aansluiting bij te vinden. Johan, die gereserveerde intelligentie en uitbundige *joie-de-vivre* weet te combineren. Wim die er altijd pit in hield. Herman, die in al z'n veelzijdigheid blijft boeien. De capabele Peter Soetens die bruist van de levenslust. Diederik en Peter Slaets, met hun grote gevoel voor humor, altijd bereid om te helpen. Tinne, die haar wiskundig kunnen combineert met emotionele intelligentie. Bedankt ook Ruben voor je teamspirit en om onze robots werkende te houden. En dan Klaas natuurlijk, waarmee ik meer gemeen heb dan we zelf willen toegeven geloof'k.
Bedankt Wim, Johan, Klaas, Peter en Peter, om me in de beginperiode op weg te zetten in deze job. Merci ook aan die andere Wim, Wim Aerts, om in een drukke periode toch de tijd te nemen om mijn tekst te lezen en allerlei suggesties te doen. Ik bedank ook m'n ouders voor hun steun. Dank ook aan alle

mensen die vandaag helpen aan de receptie en dergelijke, jullie enthousiasme is aanstekelijk.

Ringrazio anche Ducchio Fioravanti all'università degli Studi di Firenze per aver pensato con me sulla calibrazione geometrica con tanto entusiasmo.

Tot slot wil ik nog de leden van mijn jury danken voor het lezen van m'n doctoraatstekst en alle nuttige feedback.

Kasper Claes
Leuven, 27 mei 2008

Abstract

This research increases the autonomy of a **6 degree of freedom robotic arm**. We study a broadly applicable **vision** sensor, and use active sensing through projected light. Depth measurement through triangulation is based on the presence of **texture**. In many applications this texture is insufficiently present. A solution is to replace one of the cameras with a **projector**. The projector has a fixed but unknown position and the camera is attached to the end effector: through the position of the robot, the position of the camera is known and used to guide the robot.

Based on the theory of **perfect maps**, we propose a deterministic method to generate structured light patterns independent of the relative 6D orientation between camera and projector, with any desired alphabet size and **error correcting** capabilities through a guaranteed minimal Hamming distance between the codes. We propose an adapted **self-calibration** based on this eye-in-hand setup, and thus remove the need for less robust calibration objects. The triangulation benefits from the **wide baseline** between both imaging devices: this is more robust than the structure from motion approach.
The experiments show the controlled motion of a robot to a chosen position near an **arbitrary object**. This work reduces the 3D resolution, as it is anyhow not needed for the robot tasks at hand, to increase the **robustness** of the measurements. Not only using error correcting codes, but also through a robust visualisation of these codes in the projector image using only **relative intensity** values. Moreover, the projected pattern is adapted to a region of interest in the image. The thesis contains an evaluation of the mechanical **tolerances** in function of the system parameters, and shows how to control a robot with the depth measurements through **constraint based task specification**.

Beknopte samenvatting

I Vulgariserend abstract

Hoe geef je een robotarm dieptezicht?

Een robot laten zien Hoe kunnen we een robotarm de afstanden tot de verschillende punten in zijn omgeving doen inschatten? In dit werk rusten we een robotarm uit met een videocamera, en zorgen er zo voor dat de robot kan *zien* zoals een mens, zij het in een vereenvoudigde vorm. De robot kan met dit systeem niet alleen een diepteloos beeld zien, zoals een mens met één oog kan, maar krijgt ook dieptezicht, zoals een mens die beide ogen gebruikt. Door dat dieptezicht kan hij de afstand tot de voorwerpen in zijn omgeving inschatten, nodig om naar een gewenst voorwerp te kunnen bewegen. Tijdens die beweging kan de robot dan iets veranderen aan het voorwerp: bijvoorbeeld verven, lijmen of aanschroeven. Zonder dieptezicht zijn die taken onmogelijk: dan weet de robot niet hoever hij moet bewegen tot aan het voorwerp. We werken een paar van die toepassingen uit, maar houden het systeem algemeen toepasbaar.

Hoe werkt dieptezicht Wat gebeurt er als alles in de omgeving dezelfde kleur heeft, weten we dan ook hoever alles staat? Maar voor we daartoe komen eerst een algemene uitleg over dieptezicht.
Dieptezicht krijg je door dezelfde omgeving te bekijken met een tweede camera die wat verschoven staat ten opzichte van de eerste. Op die manier krijg je twee licht tegenover elkaar verschoven beelden, zoals bij de ogen van de mens. Een punt in de omgeving vormt samen met twee corresponderende punten in beide camerabeelden een driehoek. Door de hoeken en zijden van die driehoek uit te rekenen, ken je de afstand tussen de camera's en dat punt in de omgeving.

Een egaal gekleurde omgeving is problematisch Dit systeem voor dieptezicht werkt prima als er voldoende duidelijk te onderscheiden punten te zien zijn, maar niet bij een egaal gekleurde omgeving. Net zoals een mens die naar een egaal gekleurd mat oppervlak kijkt ook niet kan inschatten hoever het staat. Dat komt omdat u die driehoek niet kan vormen zoals beschreven in vorige alinea: het is niet duidelijk welk punt in het ene beeld overeenkomt met welk punt in het andere beeld.

Een projector lost het op Daarom vervangen we één van de twee camera's door een projector, zo een waar je presentaties mee zou kunnen geven. Die projecteert punten op de (eventueel egale) omgeving en zorgt zo artificieel voor punten die duidelijk te onderscheiden zijn in het videobeeld. Zo kunnen we terug een driehoek vormen tussen het beschenen punt, en de overeenkomstige punten in camera- en projectorbeeld. De kunst is de lichtpunten uit elkaar te houden in het videobeeld. De technieken die daarvoor bestaan moeten een balans zien te vinden tussen van veel punten tegelijk de diepte kunnen inschatten, en die punten betrouwbaar terug kunnen vinden. In dit werk kiezen we voor dat laatste: het is voor veel robottoepassingen minder belangrijk een gedetailleerd beeld te hebben van de omgeving, dan wel zekerheid te hebben over de gemeten afstanden.

II Wetenschappelijk abstract

Dit werk is een bijdrage in de verhoging van de autonomie van een **robotarm met 6 vrijheidsgraden**. We gaan op zoek naar een **visiesensor** die breed toepasbaar is. Meestal baseren diepteschattingen door triangulatie zich op **textuur** in de scene. In heel wat toepassingen is die textuur onvoldoende aanwezig. Een oplossing is om een van de camera's te vervangen door een projector. De projector heeft een vaste maar ongekende positie t.o.v. zijn omgeving en de camera zit vast aan de eindeffector: met behulp van de positie van de robotgewrichten kennen we de positie van de camera, dat helpt om de robot te controleren.

We stellen een deterministische methode voor om patronen te genereren onafhankelijk van de relatieve pose tussen camera en projector, gebaseerd op de theorie van **perfect maps**. De methode laat toe om een gewenste alfabetgrootte te specifiëren en een minimale Hamming-afstand tussen de codes (en biedt dus **foutcorrectie**). We stellen een aangepaste autocalibratie voor gebaseerd op deze robotconfiguratie, en vermijden daardoor het gebruik van minder robuuste calibratietechnieken op basis van een calibratieobject. De triangulatie verbetert door het gebruik van de grote **afstand tussen camera en projector**, een stabielere methode dan de diepte af te leiden uit opeenvolgende cameraposities alleen.

De experimenten laten de gecontroleerde beweging zien van de robot naar een gewenste positie tov een **willekeurig voorwerp**. De 3D resolutie is beperkt in dit werk, aangezien een hogere resolutie niet nodig is voor het uitvoeren van de taken. Dit is ten voordele van de **robuustheid** van de metingen. Die wordt niet alleen in de hand gewerkt door foutcorrigerende codes in het projectiepatroon, maar ook door een robuuste visualisatie van de codes in het projectorbeeld, alleen gebruik makende van **relatieve intensiteitswaarden**. Het projectiepatroon is ook aangepast aan het gebied in het camerabeeld dat interessant is voor de uit te voeren taak. De thesis bevat een evaluatie van de **mechanische fouten** die de robot maakt in functie van de systeemparameters, en laat zien hoe de arm kan gecontroleerd worden met behulp van het paradigma van **taakspecificatie door beperkingen op te leggen**.

Symbols, definitions and abbreviations

General abbreviations

nD	: **n**-**D**imensional
API	: **a**pplication **p**rogramming **i**nterface
CCD	: **c**harge-**c**oupled **d**evice
CMOS	: **c**omplementary **m**etal **o**xide **s**emiconductor
CMY	: {**c**yan, **m**agenta, **y**ellow} colour space
CPU	: **c**entral **p**rocessing **u**nit
DCAM	: 1394-based **d**igital **cam**era specification
DMA	: **d**irect **m**emory **a**ccess
DMD	: **D**igital **M**icromirror **D**evice
DOF	: **d**egrees **o**f **f**reedom
DLP	: **D**igital **L**ight **P**rocessing
DVI	: **D**igital **V**isual **I**nterface
EE	: **e**nd **e**ffector
EM	: **e**xpectation **m**aximisation
FFT	: **f**ast **F**ourier **t**ransform
FSM	: **f**inite **s**tate **m**achine
FPGA	: **f**ield-**p**rogrammable **g**ate **a**rray
GPU	: **g**raphical **p**rocessing **u**nit
HSV	: {**h**ue **s**aturation **v**alue} colour space
IBVS	: **i**mage **b**ased **v**isual **s**ervoing
IEEE	: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IIDC	: **I**nstrumentation & **I**ndustrial **D**igital **C**amera
KF	: **K**alman **f**ilter
LCD	: **l**iquid **c**rystal **d**isplay
LDA	: **l**inear **d**iscriminant **a**nalysis
LUT	: look**u**p **t**able
MAP	: **m**aximum **a** **p**osteriori
OROCOS	: **O**pen **Ro**bot **Co**ntrol **S**oftware
OO	: **o**bject **o**riented
OS	: **o**perating **s**ystem
PBVS	: **p**osition **b**ased **v**isual **s**ervoing

PCA	: **p**rincipal **c**omponent **a**nalysis
PCB	: **p**rinted **c**ircuit **b**oard
PDF	: **p**robability **d**ensity **f**unction
RANSAC	: **ran**dom **sa**mpling **c**onsensus
RGB	: {**r**ed **g**reen **b**lue} colour space
SLAM	: **s**imultaneous **l**ocalisation and **m**apping
STL	: **S**tandard **T**emplate **L**ibrary
SVD	: **s**ingular **v**alue **d**ecomposition
TCP	: **T**ransmission **C**ontrol **P**rotocol
UDP	: **U**ser **D**atagram **P**rotocol
UML	: **U**nified **M**odelling **L**anguage
USB	: **U**niversal **S**erial **B**us
VGA	: **V**ideo **G**raphics **A**rray
VS	: **v**isual **s**ervoing

Notation conventions

a	: scalar (unbold)
$\mathbf{a}$	: vector (bold lower case)
$\mathbf{A}$	: matrix (bold upper case)
$\|\mathbf{A}\|_\mathbf{W}$	: weighted norm with weighting matrix $\mathbf{W}$
$\mathbf{A}^\dagger$	: Moore Penrose pseudo-inverse
$\mathbf{A}^\#$	: weighted pseudo-inverse
$\|\mathbf{A}\|$	: Determinant of $\mathbf{A}$
$[\mathbf{a}]_\times$	: matrix expressing the cross product of $\mathbf{a}$ with another vector
$\|\mathbf{a}\|$	: Euclidean norm of $\mathbf{a}$

Robotics symbols

$\mathbf{J}$	: robot Jacobian
ω	: rotational speed
$\mathbf{q}$	: vector of joint positions
$\mathbf{R}$	: 3×3 rotation matrix
$\mathbf{P}$	: 3×4 projection matrix
$\mathbf{S}_b^a$	: 6×6 transformation matrix from frame a to b
${}_c\mathbf{t}_a^b$	: kinematic twist (6D velocity) of b with respect to a, expressed in frame c
t	: time
$\mathbf{v}$	: translational speed
$\mathbf{x} \equiv (x, y, z)$	: 3D world coordinate

Code theory symbols

a	: number of letters in an alphabet
b	: bit
B	: byte
H	: entropy
h	: Hamming distance

Vision symbols

c	: as subscript: indicating the camera
f	: principal distance
F	: focal length
fps	: frames per second
k	: perimeter efficiency
κ	: radial distortion coefficient
λ	: eigenvalue
p	: as subscript: indicating the projector
pix	: pixel
(ψ, θ, ϕ)	: rotational part of the extrinsic parameters
Q	: isoperimetric quotient
r, c	: respectively the number of rows and columns in the projected pattern
$\mathbf{\Sigma}$	: diagonal matrix with singular values
$\mathbf{u} \equiv (u, v)$	: pixel position, (0,0) at top left of the image
$\mathbf{U}$	: orthogonal matrix with left singular vectors
$\mathbf{V}$	: orthogonal matrix with right singular vectors
w	: width of the uniquely identifiable submatrix in the projected pattern
$\mathcal{W}_c, \mathcal{H}_c$	: respectively width and height of the camera image
$\mathcal{W}_p, \mathcal{H}_p$	: respectively width and height of the projector image

Probability theory symbols

A	: scalar random variable
$\mathbf{A}$	: vector valued random variable
H	: hypothesis
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$	: Gaussian PDF with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$)
$P(A = a)$, $P(a)$	: probability of A=a
$P(A = a\|B = b)$, $P(a\|b)$	: probability of A=a given B=b
σ	: scalar standard deviation

Table of contents

Voorwoord i

Abstract iii

Beknopte samenvatting v
I Vulgariserend abstract v
II Wetenschappelijk abstract vi

Symbols, definitions and abbreviations vii

Table of contents xi

List of figures xv

1 Introduction **1**
1.1 Scope 1
1.2 Open problems and contributions 4
1.3 Outline of the thesis 7

2 Literature survey **9**
2.1 Robot control using vision 9
2.1.1 Motivation: the need for depth information 9
2.2 3D acquisition 12
2.2.1 Time of flight 13
2.2.2 Triangulation 13
2.2.3 Other reconstruction techniques 16
2.3 Conclusion 16

3 Encoding **17**
3.1 Introduction 18
3.2 Pattern logic 19
3.2.1 Introduction 19
3.2.2 Positioning camera and projector 20
3.2.3 Choosing a coding strategy 25
3.2.4 Redundant encoding 30

3.2.5 Pattern generation algorithm . . . 33
3.2.6 Hexagonal maps . . . 37
3.2.7 Results: generated patterns . . . 39
3.2.8 Conclusion . . . 40
3.3 Pattern implementation . . . 41
3.3.1 Introduction . . . 41
3.3.2 Spectral encoding . . . 44
3.3.3 Illuminance encoding . . . 47
3.3.4 Temporal encoding . . . 49
3.3.5 Spatial encoding . . . 51
3.3.6 Choosing an implementation . . . 55
3.3.7 Conclusion . . . 57
3.4 Pattern adaptation . . . 59
3.4.1 Blob position adaptation . . . 59
3.4.2 Blob size adaptation . . . 59
3.4.3 Blob intensity adaptation . . . 59
3.4.4 Patterns adapted to more scene knowledge . . . 60
3.5 Conclusion . . . 60

4 Calibrations **61**
4.1 Introduction . . . 61
4.2 Intensity calibration . . . 62
4.3 Camera and projector model . . . 69
4.3.1 Common intrinsic parameters . . . 70
4.3.2 Projector model . . . 72
4.3.3 Lens distortion compensation . . . 73
4.4 6D geometry: initial calibration . . . 75
4.4.1 Introduction . . . 75
4.4.2 Uncalibrated reconstruction . . . 79
4.4.3 Using a calibration object . . . 80
4.4.4 Self-calibration . . . 81
4.5 6D geometry: calibration tracking . . . 97
4.6 Conclusion . . . 98

5 Decoding **99**
5.1 Introduction . . . 100
5.2 Segmentation . . . 100
5.2.1 Feature detection . . . 100
5.2.2 Feature decoding . . . 103
5.2.3 Feature tracking . . . 109
5.2.4 Failure modes . . . 110
5.2.5 Conclusion . . . 113
5.3 Labelling . . . 114
5.3.1 Introduction . . . 114
5.3.2 Finding the correspondences . . . 114
5.3.3 Conclusion . . . 116

5.4 Reconstruction . . . 117
5.4.1 Reconstruction algorithm . . . 117
5.4.2 Accuracy . . . 120
5.5 Conclusion . . . 132

6 Robot control **133**
6.1 Sensor hardware . . . 133
6.1.1 Camera . . . 133
6.1.2 Projector . . . 136
6.1.3 Robot . . . 136
6.2 Motion control . . . 137
6.2.1 Introduction . . . 137
6.2.2 Frame transformations . . . 138
6.2.3 Constraint based task specification . . . 139
6.3 Visual control using application specific models . . . 145
6.3.1 Supplementary 3D model knowledge . . . 145
6.3.2 Supplementary 2D model knowledge . . . 146

7 Software **147**
7.1 Introduction . . . 147
7.2 Software design . . . 148
7.2.1 I/O abstraction layer . . . 148
7.2.2 Image wrapper . . . 151
7.2.3 Structured light subsystem . . . 151
7.2.4 Robotics components . . . 152
7.3 Hard- and software to achieve computational deadlines . . . 155
7.3.1 Control frequency . . . 155
7.3.2 Accelerating calculations . . . 156
7.4 Conclusion . . . 158

8 Experiments **159**
8.1 Introduction . . . 159
8.2 Object manipulation . . . 160
8.2.1 Introduction . . . 160
8.2.2 Structured light depth estimation . . . 161
8.2.3 Conclusion . . . 164
8.3 Burr detection on surfaces of revolution . . . 164
8.3.1 Introduction . . . 164
8.3.2 Structured light depth estimation . . . 166
8.3.3 Axis reconstruction . . . 169
8.3.4 Burr extraction . . . 173
8.3.5 Experimental results . . . 174
8.3.6 Conclusion . . . 177
8.4 Automation of a surgical tool . . . 178
8.4.1 Introduction . . . 178
8.4.2 Actuation of the tool . . . 180

8.4.3 Robotic arm control . . . 183
8.4.4 Structured light depth estimation . . . 184
8.4.5 2D and 3D vision combined . . . 186
8.4.6 Conclusion . . . 188
8.5 Conclusion . . . 188

9 Conclusions **189**
9.1 Structured light adapted to robot control . . . 189
9.2 Main contributions . . . 191
9.3 Critical reflections . . . 192
9.4 Future work . . . 193

References **195**

A Pattern generation algorithm **205**

B Labelling algorithm **209**

C Geometric anomaly algorithms **213**
C.1 Rotational axis reconstruction algorithm . . . 213
C.2 Burr extraction algorithm . . . 216

Index **217**

Curriculum vitae **220**

List of publications **221**

Nederlandstalige samenvatting **I**
1 Inleiding . . . I
1.1 Open problemen en bijdragen . . . II
1.2 3D-sensoren . . . IV
2 Encoderen . . . V
2.1 Patroonlogica . . . V
2.2 Patroonimplementatie . . . VII
2.3 Patroonaanpassing . . . IX
3 Calibraties . . . IX
3.1 Intensiteitscalibratie . . . IX
3.2 Geometrische calibratie . . . X
4 Decoderen . . . X
4.1 Segmentatie . . . X
4.2 Etikettering . . . XI
4.3 3D-reconstructie . . . XI
5 Robotcontrole . . . XII
6 Software . . . XII
7 Experimenten . . . XII

List of Figures

1.1 The setup used throughout this thesis 2
1.2 Overview of the chapters 7

2.1 Robot control using IBVS 9
2.2 Robot control using PBVS 11

3.1 Structured light and information theory 18
3.2 Overview of different processing steps in this thesis, with focus on encoding. 19
3.3 Different eye-in-hand configurations 21
3.4 A robot arm using a laser projector 23
3.5 Conditioning of line intersections in epipolar geometry 24
3.6 Projection patterns 27
3.7 Perfect map patterns 32
3.8 Hexagonal pattern 38
3.9 Spectral implementation of a pattern 44
3.10 Selective reflection 44
3.11 Spectral response of the AVT Guppy F-033 45
3.12 Illuminance implementation of a pattern and optical crosstalk 48
3.13 Temporal implementation of a pattern: different frequencies . 49
3.14 Temporal implementation of a pattern: different phases . . . 50
3.15 1D binary pattern proposed by Vuylsteke and Oosterlinck . . 51
3.16 Shape based implementation of a pattern 52
3.17 Spatial frequency implementation of a pattern 53
3.18 Spatial frequency implementation: segmentation 54
3.19 Concentric circle implementation of a pattern 56

4.1 Overview of different processing steps in this thesis, with focus on calibration 62
4.2 Reflection models 62
4.3 Monochrome projector-camera light model 63
4.4 Camera and projector response curves 67
4.5 Pinhole model compared with reality 71
4.6 Upward projection 72

4.7 Asymmetric projector opening angle 73
4.8 Pinhole models for camera - projector pair 73
4.9 Projector-camera geometric calibration vs structure from motion 75
4.10 Angle-side-angle congruency 76
4.11 Frames involved in the triangulation 77
4.12 Calibration of camera and projector using a calibration object 80
4.13 Self-calibration vs calibration using calibration object 81
4.14 Crossing rays and reconstruction point 82
4.15 Furukawa and Kawasaki calibration optimisation 83
4.16 Cut of the Furukawa and Kawasaki cost function 84
4.17 Epipolar geometry 85

5.1 Overview of different processing steps in this thesis, with focus on decoding 100
5.2 Camera image of a pattern of concentric circles 101
5.3 Standard deviation starting value 104
5.4 Automatic thresholding without data circularity 104
5.5 Automatic thresholding with mirroring 105
5.6 Difference between prior and posterior threshold 105
5.7 Identification prior for relative pixel brightness 107
5.8 Validity test of local planarity assumption 112
5.9 Labelling experiment 116
5.10 Ray - plane intersection conditioning 118
5.11 Coplanarity assumption of accuracy calculation by Chang . . 120
5.12 Contribution of pixel errors for the principal point 123
5.13 Contribution of pixel errors for $\psi = \frac{\pi}{2}$ 123
5.14 Side views of the function $\mathcal{E}$, sum of squared denominators . . 124
5.15 Assumption of stereo setup for numerical example 125
5.16 Error contribution of the camera principal distance 127
5.17 Error contribution of the projector principal distance 128
5.18 Cut of the second factor of equation 5.13 in function of ψ and θ 128
5.19 Error contribution of the frame rotation: θ 131
5.20 Error contribution of the frame rotation: ϕ 132

6.1 Overview of the hardware setup 134
6.2 Frame transformations 138
6.3 Positioning task experiment setup and involved frames 140
6.4 Frame transformations: object and feature frames 142

7.1 UML class diagram 149
7.2 FSM 154
7.3 Accelerating calculations using a hub 158

8.1 Experiment setup 160
8.2 3D reconstruction result 163

8.3 Robot arm and industrial test object 164
8.4 Local result of specularity compensation 167
8.5 Global result of specularity compensation 168
8.6 A structured light process adapted for burr detection 168
8.7 Determining the orientation of the axis. 170
8.8 Uniform point picking 171
8.9 Axis detection result 171
8.10 Mesh deviation from ideal surface of revolution 172
8.11 Determining the burr location 173
8.12 Error surface of the axis orientation 174
8.13 Configurations corresponding to local minima 175
8.14 Quality test of the generatrix 176
8.15 Unmodified Endostitch 178
8.16 Detailed view of the gripper and handle of an Endostitch . . 180
8.17 Pneumatically actuated Endostitch 181
8.18 State machine and robot setup 182
8.19 Minimally invasive surgery experiment 183
8.20 Setup with camera, projector, calibration box and mock-up . 184
8.21 High resolution reconstructions for static scene 185
8.22 Structured light adapted for endoscopy 186
8.23 2D and 3D vision combined 188

9.1 Overview of different processing steps in this thesis 191

A.1 Overview of dependencies in the algorithm methods 205

1 De opstelling die doorheen de thesis bestudeerd wordt II
2 Patroonimplementatie op basis van concentrische cirkels . . . VIII
3 Overzicht van de verschillende stappen voor 3D-reconstructie XI

Chapter 1

Introduction

Caminante, no hay camino,
se hace camino al andar.

Antonio Machado

1.1 Scope

Why one needs sensors

Still today, robotic arms mostly **use proprioceptive sensors only**. Proprioceptive sensors are sensors that enable the robot to detect the position of its own joints. In that case, the information available to the robot programmer is limited to the relative position of parts of the arm: this is a blind, deaf, and numb robot. This poses no problem if the position of the objects in the environment to be manipulated is known exactly at any point in time. It allows for an accurate, fast and uninterrupted execution of a repetitive task.

However, the use of **exteroceptive sensors** (e.g. a camera) enables the robot to **observe** its environment. Thus, the robot does not necessarily need to know the position of the objects in its environment beforehand, and the environment can even be changing over time. Exteroceptive sensors have been studied in the academic world for decades, but only slowly find their way to the industrial world, since the interpretation of the information of these sensors is a complex matter.

Assumptions

The volume of the environment of the robot is in the order of $1m^3$. The robot has enough joints and hence liberty of movement such that an object attached to it can be put in any position within the reach of the robot. We say the robot has **6** (motion) **degrees of freedom**: the hand of the robot can translate in 3 directions and rotate in 3 directions. Hence, we do **not** assume a **planar** environment.

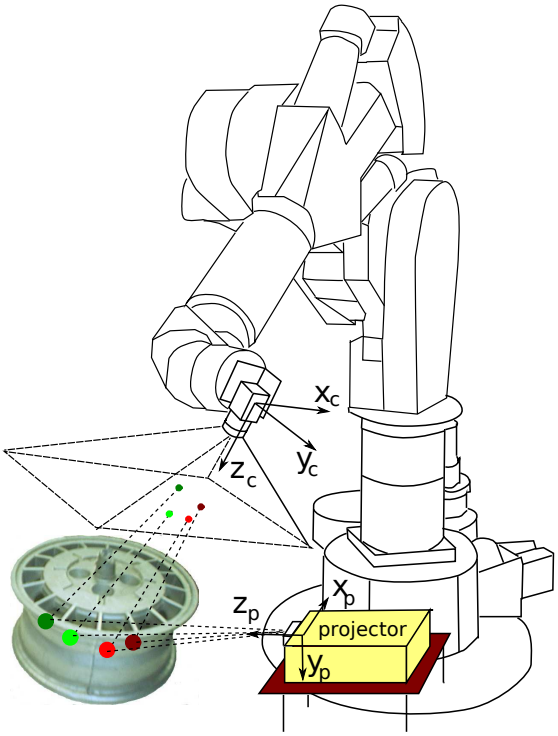


Figure 1.1: The setup used throughout this thesis

Using computer vision

The exteroceptive sensor studied in this thesis is a camera. Positioning a robot with vision depends on recognisable **visual features**: one cannot navigate in an environment when everything looks the same. Such visual features are **not always present**: for example when navigating along a uniformly coloured wall. There is a **solution** to this problem: this thesis uses **structured light**, a subfield of computer vision that uses a projection device alongside a camera. Together with the environment, the projector controls the visual input of the camera. Structured light solves the problem of the possible lack of visual features by projecting artificial visual features on the scene. The **aim** of projecting these features is to be able to **estimate the distance** to objects. Figure 1.1 shows this setup. The features are tracked by the camera on the robot, and incrementally increase the robot's knowledge about the scene. This 3D reconstruction uses no more than – inexpensive – consumer grade hardware: a projector and a camera.

The **reconstruction** is not a target on itself, but **a means** to improve the capabilities of the robot to execute tasks in which this depth information is useful. Hence often, the system does not reconstruct the full scene but only determines the depth for those parts that are needed for the robot task at hand.

The 3D resolution is minimal, as low as the execution of the robot task allows: all **additional resolution is a waste of computing power**, as the measurement are to be directly used online. The feature recognition initialisation does not have to be repeated at every time step: the features can be **tracked**. The thesis emphasises robustness, as the **environment** is often **not conditioned**. This contrasts with dense 3D reconstruction techniques where the aim is to precisely inverse engineer an object.

Applications

Many robot tasks can benefit from structured light, but we emphasise those that are hard to complete without it: tasks with a lack of natural visual features. We discuss two types of applications: **industrial** and **medical** ones. For example when painting industrial parts, structured light is useful to detect the geometry of the object and calculate a path for the spray gun. See section 8.2 for a discussion on practical applications.
Often human organs also have little natural features, so this structured light technique is useful in endoscopic surgery too to estimate the distances to parts of the organ, see section 8.4 [Hayashibe and Nakamura, 2001].
Many applications will benefit from not only using a vision sensor, but by integrating cues from various kinds of sensory input.

1.2 Open problems and contributions

Even after more than a quarter century research on structured light [Shirai and Suva, 2005], some issues remain:

- Problem: Previous applications of structured light in robotics have been limited to static camera and projector positions, and static scenes: reconstruct the scene, and then move towards it [Jonker et al., 1990]. Recently [Pagès et al., 2006] the first experiments were done with a camera attached to the end effector and a single shot pattern. For other applications of structured light than robotics, one keeps the **relative position between camera and projector** constant. This leads to less complex mathematics to calculate the distances that separate the camera from its environment than if the relative position would not be constant. However, the latter is necessary: section 3.2.2 explains why a configuration where the projector cannot be moved around is needed to control a robot arm and why the camera is attached to the end effector.
  Pagès et al. [2006] has been the first to work with this changing relative position. However, he does not use the full mathematical potential of this configuration: he does not calibrate the projector-camera pair. Calibrating here means estimating the relative position between camera and projector, and taking advantage of this knowledge to improve the accuracy of the result.
  Contribution: Incorporate this **calibration** for a **baseline** that is **changing** online, and thereby **improving** the reconstruction **robustness** (see section 4.4).

- Problem: Normally projector and camera are oriented in the same direction: what is up, down, left and right in the projector image remains the same in the camera image. However, the camera at the hand of the robot cannot only translate in 3 directions, but also rotate in 3. Salvi et al. [2004] give an overview of structured light technique throughout the last quarter century: all rely on this known relative rotation between camera and projector, usually hardly rotated.
  Contribution: Novel in this work is the **independence of the pattern of the relative rotation between camera and projector**. In section 3.2.3.2 a technique is presented to generate patterns that comply to this independence.

- Problem: Until recently structured light studied the reconstruction of a point cloud of a static object only, as it was necessary to project several images before the camera had retrieved all necessary information. This is called temporal encoding. During last years, it also became possible to estimate depths using structured light *online*: see for example [Adán et al., 2004], [Chen et al., 2007], [Hall-Holt and Rusinkiewicz, 2001], [Vieira et al., 2005] and [Koninckx et al., 2003]. Hall-Holt and Rusinkiewicz [2001] and Vieira et al. [2005] use temporal encoding techniques that need very little

image frames. These techniques can therefore work online, as long as the motion is slow enough compared to (camera) frame rate.
Adán et al., Chen et al. and Koninckx et al. do use single-shot techniques: the camera retrieves all information in a single camera image. Therefore the speed of objects in the scene is not an issue with these methods [1].
However, all these techniques depend on colours to function, and will hence fail with some coloured scenes, unless one adds additional camera frames again to adapt the projector image to the scene. But then, the technique is not single-shot any more. In robotics one often works with a moving scene, hence a single-shot technique is a necessity.
Contribution: The technique we propose is also sparse as are the techniques of for example Adán et al., Chen et al. and Koninckx et al., but does not depend on colours. It is **based on the relative difference in grey levels**, and hence is truly **single-shot and independent of local surface colour or orientation**.

- Problem: In structured light, there always is a **balance** between robustness and 3D resolution. The finer the projected resolution, the larger the chance of confusing one projected feature with another. In this work, the image processing on the observed image focuses on the interpretation of the scene the robot works in. Thus for control applications, a relatively **low resolution** suffices, as the movement can be corrected iteratively online, during the motion. When the robot end effector is still far away from the object of interest, a course motion vector suffices to approach the object. At that point the robot does not need a high 3D resolution. As the end effector, and hence also the camera moves closer, the images provide us with a more detailed view of the object: the projected features appear larger. Hence we can afford to make the features in the projector image smaller, while keeping the size of those features in the camera image the same, and hence not increasing the chance of confusing one projection feature with another. This process is of course limited by the projector resolution.
  Contribution: We choose for a low 3D resolution but high robustness. As this resolution is dependent on the position of the robot, we **zoom** in or out **in the projector image** accordingly, to **adapt the 3D resolution online** according to the needs, for example to keep it constant while the robot moves (see section 3.4).

- Problem: Often, **depth discontinuities** cause occlusions of pattern features. Without error correction, one cannot reconstruct the visible features near a discontinuity, as the neighbouring features are required to associate the data between camera and projector. Also, when part of a feature is occluded such that it cannot be correctly segmented any longer, that feature is normally lost. Unless error correction can reconstruct how the feature would have been segmented if it were completely visible.
  Contribution: Because of the low resolution (see previous contribution),

[1] That is, not taking in account motion blur (see section 3.3.7).

we can afford to increase the redundancy in the projector image, to improve robustness, in a way orthogonal to the previous contribution. More precisely, we add error correcting capabilities to the projector code. The code is such that the projected image does not need more intensity levels than other techniques, but if one of the projected elements is not visible, that error can be corrected. Adán et al. [2004] and Chen et al. [2007] for example do not provide such error correction.
We provide the projected pattern with **error correction**. The resolution of the pattern is higher than similar techniques with error correction [Morano et al., 1998] for a constant number of grey levels. However, the constraints on the pattern are more restrictive: it has to be independent of the viewing angle (see above). In other words, for a constant resolution, our technique is capable of **correcting more errors** (see chapter 3.2).

- Problem: Many authors do not make an explicit difference between the code that is incorporated in the pattern, and the way of projecting that code. The result is that some of the possible combinations of both remain unthought of.
  Contribution: This thesis separates the methods to generate the logic of abstract patterns (section 3.2), and the way they are put into practice (section 3.3): these two are orthogonal. It **studies a variety of ways to implement the generated patterns**, to make an explicitly motivated decision about what pattern images are most suited for applications with a robot arm.

- Problem: How to use this point cloud information to perform a task with a robot arm?
  Contribution: We **apply** the techniques of **constraint based task specification** to this structured light setup: this provides a mathematically elegant way of specifying the task based on 3D information, and allows for an easy integration with data coming from other sensors, at other frequencies (see section 6.2).

- Problem: the uncertainty whether the camera-projector calibration and the measurements are of sufficient quality to control the robot arm, in the correct direction within the specified **geometric tolerances**.
  Contribution: an **evaluation** of the **mechanical errors**, based on a projector and a camera that can be positioned anywhere in a **6D** world. This is a high dimensional error function, but by making certain well considered assumptions, it becomes clear which variables are sensitive in which range, and which are more robust (see section 5.4.2).

1.3 Outline of the thesis

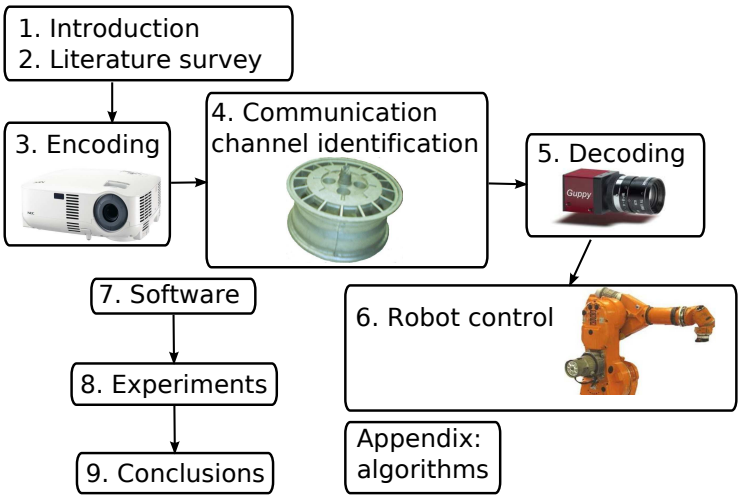


FIGURE 1.2: Overview of the chapters

Figure 1.2 presents an overview of the different chapters:

1. This chapter introduces the aim and layout of this thesis.

2. Chapter 2 places structured light in a broader context of **3D sensors**. For example, section 2.1 discusses how one can achieve motion control of a robot arm: what input data does one need.

3. Chapters 3 and 5 discuss the **communication** between projector and camera: they cooperate to detect the depth of points in the scene. Communication implies a communication language, or code: chapter 3 discusses the formation of this code at the projector side, chapter 5 elaborates on its decoding on the camera side.
   The **encoding** chapter (chapter 3) contains three sections about the creation of this code:

   - Section 3.2 discusses the **mathematical properties** of the projected pattern as required for reconstructing a scene.
   - These mathematical properties are independent of the practical **implementation** of the projected features, see section 3.3.
   - **Active sensing**: the projected pattern is not static. A projector provides the freedom to **change the pattern** online: the size, position and brightness of these features are altered as desired to gain better knowledge about the environment, see section 3.4.

4. To decode the projected pattern, one needs to estimate a number of parameters that characterise the camera and projector. This is called calibration. Several types of calibration are needed, all of which can be done automatically. In chapter 4, we calibrate the sensitivities to light intensity, take lens properties into account and determine the parameters involved in the 6D geometry.

5. Also the decoding chapter (chapter 5) has three sections about interpreting the received data:

   - The reflection of the pattern together with the ambient light on the scene is perceived by the camera. The image processing needed to decode every single projection feature is explained is section 5.2. The emphasis in this chapter is on algorithms that remain as independent of the scene presented as possible. Or in other words, the emphasis is on automated **segmentation** in a broad range of circumstances. This is at the expense of computational speed.
   - After segmentation, we study whether the relative position of features in the camera image is in accordance with the one in the projector image. This information is used to increase or decrease the belief in the decoding of each feature. Section 5.3 covers this **labelling** procedure.
   - Once the correspondence problem is solved and the system parameters have been estimated, one can calculate the actual reconstruction in section 5.4. This section includes an evaluation of the geometric errors made as a function of the errors on the different parameters and measurements.

6. Chapter 6 elaborates on the motion control of the robot arm, and its sensory hardware.

7. The chapters that follow describe the more practical aspects. Chapter 7 discusses the hard- and software design.

8. Chapter 8 explains the robotics experiments. The first experiment deals with general manipulation using the system described above. The second one elaborates on an industrial application: deburring of axisymmetric objects, and the last one is a surgical application: the automation of a suturing tool.

9. Chapter 9 concludes this work.

Chapter 2

Literature survey

2.1 Robot control using vision

This thesis studies the motion control of the joints of a robotic arm using a video camera. Since a camera is a 2D sensor, the most straightforward situation is to control a two degree of freedom (2DOF) robot with it. An example of a 2DOF robot is a XY table. The camera moves in 2D, and observes a 2D scene parallel to the plane of motion (the image plane). Hence, mapping the pixel coordinates to real world coordinates is trivial: this is two-dimensional control. Usually, a control scheme uses not only feedforward but also feedback control. If it uses a feedback loop, one can speak of **two-dimensional visual servoing** (2DVS).

2.1.1 Motivation: the need for depth information

If the scene is non-planar, a camera can also be used as a **3D sensor**. This is the case this thesis studies. Different choices can be made in 3D visual servoing, two main techniques exist: control in the (2D) image space (image based visual servoing) or in the 3D Cartesian space (position based visual servoing). Both have their advantages and disadvantages, as Chaumette [1998] describes. What follows is a summary including the combination of both techniques:

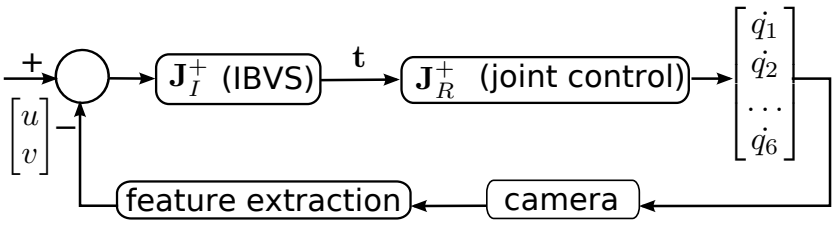


FIGURE 2.1: Robot control using IBVS

- **Image based visual servoing** (IBVS): Figure 2.1 presents this approach. A Jacobian is a matrix of partial derivatives. The image Jacobian $\mathbf{J}_I$ relates the change in image feature coordinates $(\dot{u}, \dot{v})$ to the 6D end effector

velocity, or twist. The rotational component of this velocity is expressed in the angular velocity $\boldsymbol{\omega}$: $\mathbf{t} = [\dot{x}\ \dot{y}\ \dot{z}\ \omega_x\ \omega_y\ \omega_z]^T$. The robot Jacobian $\mathbf{J}_R$ relates the 6D end effector speed with the rotational joint speeds $\dot{q}_i$ $i = 1..6$ (see section 6.2 for more details):

$$\left.\begin{aligned} \begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} &= \mathbf{J}_I\mathbf{t} \\ \mathbf{t} = \mathbf{J}_R \begin{bmatrix}\dot{q_1} & \dot{q_2} & \dots & \dot{q_6}\end{bmatrix}^T & \end{aligned}\right\} \Rightarrow \begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \mathbf{J}_I\mathbf{J}_R\dot{\mathbf{q}} \Rightarrow \dot{\mathbf{q}} = (\mathbf{J}_I\mathbf{J}_R)^\dagger \begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix}$$

For a basic pinhole model for example, with f the principal distance of the camera:

$$\mathbf{J}_I = \begin{bmatrix} \frac{f}{z} & 0 & \frac{-u}{z} & \frac{-uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{z} & \frac{-v}{z} & \frac{-f^2-v^2}{f} & \frac{uv}{f} & u \end{bmatrix}$$

Note that the term *Jacobian* is defined as a matrix of partial derivatives only, and does not specify which variables are involved. In this case for example, the twist can be the factor with which the Jacobian is multiplied (as is the case for $\mathbf{J}_I$) or it can be the result of that multiplication (the case for $\mathbf{J}_R$). The variables involved are such that the partial derivatives can be calculated. For the image Jacobian for example, $\partial u/\partial x$ can be determined, as the mapping from (x, y, z) to (u, v) is a mapping from a higher dimensional space to a lower dimensional one (hence e.g. $\partial x/\partial u$ cannot be determined). For the robot, the situation is slightly different, as both the joint space and the Cartesian space are 6 dimensional. However, the mapping from joint space to Cartesian space (forward kinematics) is non-linear, because of the trigonometric functions of the joint angles. Therefore, there is an unambiguous mapping from joint space to Cartesian space, but not the other way around (the inverse of a combination of trigonometric functions?). For example, one can express $\partial x/\partial q_1$, but not $\partial q_1/\partial x$. When using these image or robot Jacobians in a control loop, one needs their generalised inverses.

IBVS has proved to be robust against calibration and modelling errors: often all lens distortions are neglected for example, as it is sufficiently robust against those model errors. Control is done in the image space so the target can be constrained to remain visible. However, IBVS is only locally stable, so path planning is necessary to split a large movement up in smaller local movements. Also, rotation and translation are not decoupled, so planning pure rotation or translation is not possible. Moreover, IBVS suffers from servoing to local minima. And as the end effector trajectory is hard to predict, the robot can reach its joint limits. More recently, Mezouar and Chaumette [2002] proposed a method to avoid joint limits with a path planning constraint. Also, IBVS is not model-free: the model required is the image coordinates of the chosen features in the target position of the robot.

- **Position based visual servoing** (PBVS) decouples rotation and translation and there is global asymptotic stability if 3D estimation is sufficiently accurate. Global asymptotic stability refers to the property of a controller to stabilise the pose of the camera from any initial condition. However, analytic proof of this is not evident, and position based servoing does not provide a mechanism for making the features remain visible. Kyrki et al. [2004] propose a scheme to overcome the latter problem. Also, errors in calibration propagate to errors in the 3D world, so one needs to take measures to ensure robustness.
  The model required is a 3D model of the scene.

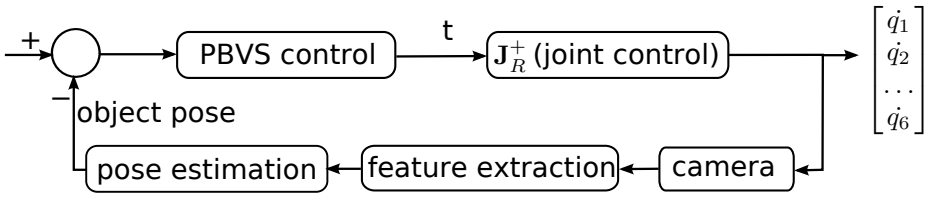


FIGURE 2.2: Robot control using PBVS

- $\mathbf{2\frac{1}{2}}$ **D** servoing (2.5DVS) combines control in image and Cartesian space [Malis et al., 1999]. These hybrid controllers try to combine the best of both worlds: they decouple rotation and translation and keep the features in the field of view. Also, global stability can be ensured. However, also here the Cartesian trajectory is hard to predict.
  The model required is the image coordinates of the chosen features in the target position.

An overview of advantages and disadvantages of these techniques can be found in table 2.1.
Usually, the positions of the features in IBVS and 2.5DVS are obtained by moving to the target position, taking an image there and then moving back again to the initial position. Needing to have an image in the target position imposes similar restrictions as the need to have a 3D model of the target for position based servoing. Hence, all techniques need model knowledge. Also, a 3D model does not always need to be a detailed CAD model. The 3D point cloud could also be fitted to a more simple 3D shape in a region of interest of the image. Hence, the need for a 3D model in PBVS is an acceptable constraint, therefore **this work focuses on PBVS** (see section 6.2).

Note that all these techniques need an explicit estimate of the depth of the feature points. Recently Benhimane and Malis [2007] proposed a new image based technique that uses no 3D data in the control law: all 3D information is contained implicitly in a (calibrated) homography. This technique is only proven to be locally stable. Local stability implies that the system can track a path,

	IBVS	*PBVS*	*2.5DVS*
robust against calibration errors	+	-	+-
target always visible	+	- / +[1]	+
independent of 3D model	+	-	+
independent of target image	-	+	-
avoids joint limits	- / +[2]	+	+-
global control stability	-	+-	+
rotation/translation decoupled	-	+	+
no explicit depth estimation	- / +[3]	-	-

[1]:[Kyrki et al., 2004], [2]: [Mezouar and Chaumette, 2002], [3]: [Benhimane and Malis, 2007]

Table 2.1: Overview of advantages and disadvantages of some visual servoing techniques

but does not necessarily remain stable for large control differences. Hence, path planning in the image space would be necessary for wide movements. The difference between this technique and previous visual servoing techniques is similar to the difference between self-calibration and calibration using a calibration object, in the sense that the former uses implicit 3D information, and the latter uses it explicitly. Section 4.4 elaborates on these calibration differences, see figure 4.13. Here we concentrate on the standard techniques that **need** explicit **depth** information. Section 2.2 elaborates on the different ways to obtain this 3D information.

2.2 3D acquisition

Blais [2004] gives an overview of non-contact surface acquisition techniques over the last quarter century. Table 2.2 summarises these technologies, only mentioning the reflective techniques, not the transmissive ones like CT.

time-of-flight		*triangulation*		*other*
acoustic	EM	active	passive	
sonar	radar lidar	laser point/line projector: -time coding -spatial coding	struct.from motion (binocular) stereo	interferometry shape from... -silhouettes -(de)focus -shading -texture

Table 2.2: Overview of reflective shape measurement techniques

2.2.1 Time of flight

One group of techniques uses the time-of-flight principle: a detector awaits the reflection of an emitted wave. This wave can be acoustic (e.g. sonar) or electromagnetic. In the latter group, radar (**ra**dio **d**etection **a**nd **r**anging) uses long wavelengths for distant objects. Devices that use shorter wavelengths, usually in the near infrared, are called lidar devices (**li**ght **d**etection **a**nd **r**anging), see [Adams, 1999]. Until recently pulse detection technology was too slow for these systems to be used at closer range than $\pm 10m$. These electronics need to be able to work at a high frequency to detect the phase difference in the very short period of time light travels to the object and back. Applications are for example in the reconstruction of buildings. Lange et al. [1999] at the CSEM lab proposes such a system.
Oggier et al. [2006] explain improvement in recent years that led to a commercially available product from the same lab that does allow working at close range (starting from $30cm$): the SwissRanger SR3000[1]. Its output is range data at a resolution of 176×144 and a frame rate of $50fps$. It is sufficiently small and light to be a promising sensor for the control of a robotic arm: phase aliasing starts at 8m, which is more than far enough for these applications (price in 2008: ± 5000 €). Gudmundsson et al. [2007] discuss some drawbacks: texture on the surfaces influences the result leading to an error of several cm.

2.2.2 Triangulation

using cameras

A second group of techniques triangulate between the position of two measurement devices and each of the points on the surface to be measured, see e.g. [Curless and Levoy, 1995]. If the relative 6D position between the two measurement devices is known, the geometry of each of the triangles that are formed between them and each of the points in the scene can be calculated. One needs to know the precise orientation of the ray between each of the imaging device and each of the visible points. That means that for each point in one measurement device the corresponding point in the other device needs to be found. In binocular stereopsis for example, both measurement devices are cameras, and the slightly shifted images can be combined into a disparity map that contains the distances between the correspondences. Simple geometrical calculations then produce the range map: the distances between the camera and the 3D point corresponding to each pixel. Another possibility is to use three cameras for stereo vision: trinocular stereopsis, in order to increase the reliability.
Instead of using two (or three) static cameras, one can also use the same stereo principle with only one moving camera: the two cameras are separated in time instead of in space. As we want to reconstruct the scene as often as possible, usually several times a second, the movement of the camera in between the transmission of two frames is small compared to the distance to the scene. The

[1] www.mesa-imaging.ch

calculation of the height of these acute triangles with two angles of almost $\frac{\pi}{2}$ is poorly conditioned. This often requires a level of reasoning above the triangulation algorithm, to filter that noisy data, for example statistically. Thus triangulation techniques suffer from bad conditioning when the baseline is small compared to the distance to the object. Time-of-flight sensors do not have this disadvantage and can also be used for objects around $100m$ away. However, in the applications studied in this thesis, only distances in the order of $1m$ are needed, a range where triangulation is feasible.

A hand-eye calibration is the estimation of the 6 parameters that define the rigid pose between the end effector frame and the camera frame. If the motion of the robot is known, and hence also the motion of the camera after performing a **hand-eye calibration**, only the position of parts of the scene has to be estimated, see for example [Horaud et al., 1995]. However, if the motion of the camera is unknown (it is for example moved by a person instead of a robot), then there is a double estimation problem: the algorithms have to estimate the 6D position of the camera too. This problem can be solved using SLAM (Simultaneous Localisation and Mapping). Davison [2003] presents such a system: online visual SLAM, using **S**hi-**T**omasi-**K**anade-features [Shi and Tomasi, 1994] to select what part of the scene is to be reconstructed. The result is an online estimation of the position of the camera and a sparse reconstruction of its environment. To process the data, he uses an information filter: the dual of a Kalman filter. The information matrix is the inverse of the covariance matrix, but from a theoretical point of view, both algorithms are equivalent. Practically, the information filter is easier to compute, as the special structure of the SLAM problem can enforce sparsity on the information matrix, reducing the complexity to O(N). This sparsity makes a considerable difference here, as the state vector is rather large: it is a combined vector with the parameters defining the camera, and the 3D positions of all features of interest in the scene. As features of interest Davison chooses the well conditioned STK-features. The parameters defining the camera are its pose, for easier calculations in non-minimal form: a 3D coordinate and quaternions, 7 parameters; and 6 for the corresponding linear and angular velocity. In other words, this motion model assumes that on average the velocities, not the positions, remain the same. Incorporating velocity parameters leads to a smoother camera motion: large accelerations are unlikely. Splitting the problem up in a prediction and a correction step allows to write the equations *from* 3D data *to* 2D projection. That problem is well behaved, hence one avoids the poor conditioning of the inverse problem: triangulation. Triangulation estimates the 3D world from a series of 2D projections with small baselines in between them: a poorly conditioned problem.

Nister et al. [2004] has a somewhat different solution to the same problem. A less important difference is his different choice of low level image feature: Harris corners. More important is that these features are tracked and triangulated between the first and the last point in time each feature is observed (structure from motion), thus minimising the problem of the bad conditioning of triangulation

with small baselines. For the other estimation problem (the camera pose), the 5-point pose estimation algorithm is used [Nister, 2004]. Together this leads to a system capable of visual odometry.

Both usefulness and complexity of stereo vision can be increased allowing the cameras to make rotational movements like human eyes, using a pan tilt unit. We will not consider this case in this thesis.

0D and 1D structured light

Solving the correspondence problem can be hard when surfaces have a sufficient amount of texture, to impossible for textureless surfaces. This data association problem is where the distinction between active and passive techniques comes into play. Active techniques project light onto the scene to facilitate finding the correspondences: one of the measurement devices is a light emitting device and one a light receiving device (a camera). In its simplest form, the light emitting device can be a laser pointer that scans the surface, highlighting one point on the surface at a time, and thus indicating the correspondence to be made. Compare this to a cathode ray tube that scans a screen (a CRT screen). Stereo vision on the other hand is a passive technique since the observed scene is not altered to facilitate the data association problem.

The projection of a single ray of laser light, is a point. Hence the name 0D structured light. It would speed things up if several points could be reconstructed at once. Therefore, as section 3.2.1 will explain further, this ray of light is usually replaced by a plane of light. This plane of laser light intersects with the surface in a line-like shape: 1D structured light. It is a technique often used in industry these days, Xu et al. [2005] for example weld in this way.

2D structured light

Projecting a **pyramid** of light is another possibility. Appoximating the light source as a point, from which the light is blocked everywhere except through the rectangular projector image plane, gives the illumination volume a pyramidal shape. Maas [1992] uses an overhead projector with a fixed grating. The grating helps to find corresponding points in multiple cameras, between which Maas triangulates. But the correspondence problem is still difficult, since all corner points of the grating are the same: the identification of the corresponding points still depends on the scene itself and not on the projected pattern.
This is comparable to the work of Pagès et al. [2006]: he also triangulates using only camera views, helped by a projector. In both works the projector is only used to help find the correspondences: no triangulation is done between camera and projector. Differences are that Pagès et al. uses different viewpoints of only one camera instead of several static cameras. In his technique, every projector feature is uniquely identifiable, which is not the case in the work by Maas.

Proesmans et al. [1996] presents work similar to [Maas, 1992], but does not determine corresponding points between the images. The reconstruction is based only on the deformation of the projected grating. The problem however with this

approach is that the reconstructed surfaces must be continuous, like the human faces presented in his experiments. Discontinuities in depth remain undetected.

Later, the overhead projector was replaced by a data projector, adding the (potential) advantage of changing the pattern during reconstruction. The projected pattern needs to be such that the correspondences can easily be found. This can be done by projecting several patterns after one another (time coding), or by making all features in the pattern uniquely identifiable (spatial coding). The latter technique is the one used in this thesis.

An advantage of structured light is that **motion blur is less of a problem** than with passive techniques, since the features projected are brighter than the ambient light, and thus the exposure time of the camera is correspondingly smaller.

2.2.3 Other reconstruction techniques

Other techniques for 3D reconstruction exist. An overview :

- The earliest attempts to reconstruct 3D models from photos used **silhouettes** of objects. Silhouettes are closed contours that form the outer border of the projection of an object onto the image plane. This technique assumes the background can be separated from the foreground. The intersection of silhouette cones taken from different viewpoints provide the visual hull: the 3D boundaries of the object. Disadvantages are that this approach cannot model concavities and the need for a controlled turntable setup with an uncluttered background [Laurentini, 1994].
- Two images of a scene obtained from the same position but using different **focal** settings of the camera also contain enough information to compute a depth map, see [Nayar, 1989].
- The amount of **shading** on the scene is another cue to determine the relative orientation of that surface patch. Different methods exist to retrieve shape from shading, see for example [Zhang et al., 1994].
- The deformation of texture on a surface can also be used to infer depth information: shape from **texture**, see [Aloimonos and Swain, 1988].
- Moiré interferometry is an example of the use of **interferometry** to reconstruct depth. A grating is projected onto the object, and from the interference pattern between the surface and another grating in front of the camera, depth information can be extracted.

2.3 Conclusion

This chapter studied the needs to control a robot arm visually. Section 2.1 discusses different strategies for this control, and mentions the relevant authors. It concludes that all these strategies are dependent on depth information. Section 2.2 then discusses how one can retrieve this 3D information. Different techniques and corresponding authors are discussed. The rest of this thesis will concentrate on structured light range scanning.

Chapter 3

Encoding

Less is more

Robert Browning, Mies van der Rohe

The projector and the camera communicate through structured light. This chapter treats all aspects of the construction of the communication code the projector sends. The important aspects are threefold:

- The mathematical properties of the code, discussed in section 3.2. This is comparable to the grammar and spelling of a language.
- The practical issues of transferring the code, discussed in section 3.3. Compare this to writing or speaking a language.
- The adaptation of the code according to the needs, discussed in section 3.4. This is comparable to reading only the parts of a book that one is interested in, or listening only to a specific part of the news.

3.1 Introduction

A way of perceiving a structured light system is as a noisy communication channel between projector and camera: the projector is the sender, the air through which the light propagates and the materials on which it reflects the channel, and the camera the receiver. This is similar to for example fiber-optic communication. The scene is also part of the sender, as it adds information.
Therefore, **information theory** can be applied to a structured light system and we will do so throughout this thesis. Figure 3.1 illustrates this: the message we want to bring across the communication channel is the 3D information about the scene. In order to do so, we multiplex it on the physical medium with the pattern, and possibly also with – unwanted – other light sources. Section 3.2 develops the communication code, and section 3.3 will explain how to implement this code: the advantages and disadvantages of different visual features to reliably transfer the information.
After this encoding follows a decoding phase. Chapter 5 decodes the information from the camera image. First the demultiplexing of the other light sources: this removes all visual features that are not likely to originate from the projector. The next demultiplexing step extracts the readable parts of the pattern, and can thereby also reconstruct the corresponding part of the scene.
Then there is feedback from the detected pattern to the projected pattern: the *pattern adaptation* arrow indicates what is explained in section 3.4: the brightness, size and position of the projected features can be adapted online.

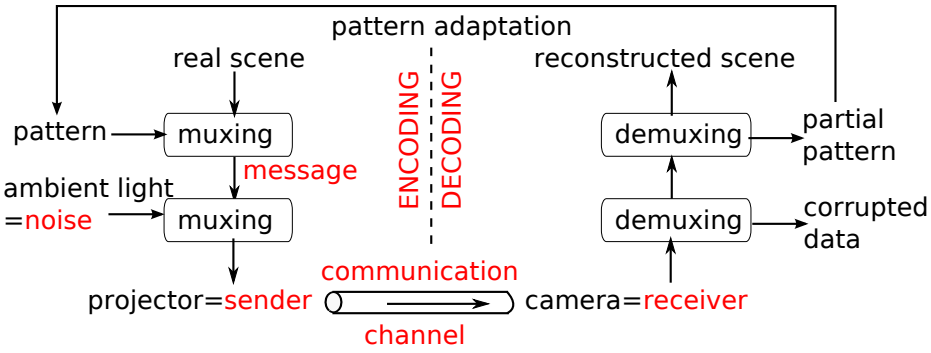


Figure 3.1: Structured light as communication between projector and camera

In terms of information theory, the fact that low resolutions are sufficient for these robotic applications leaves room to increase the **redundancy** in the projector image. This improves the robustness of the communication channel: one adds error correcting capabilities [Shannon, 1948]. However, a channel always has its physical limitations, and a compromise between minimal information loss and bandwidth imposes itself. This brings us back to the balance between robustness and 3D resolution, as introduced in chapter 1.

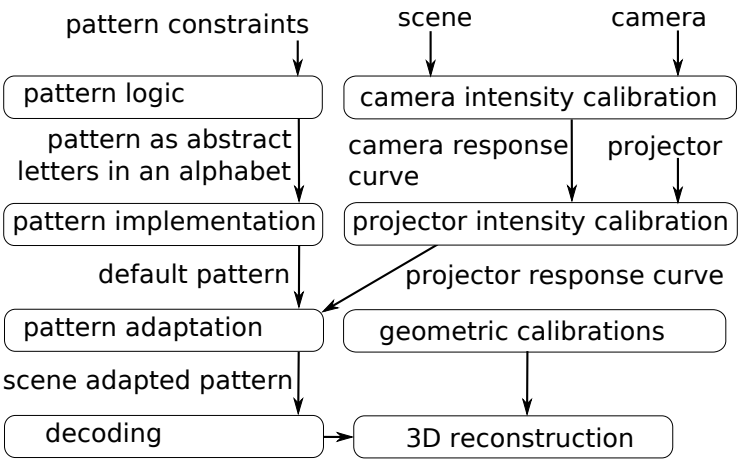


FIGURE 3.2: Overview of different processing steps in this thesis, with focus on encoding.

Figure 3.2 presents an overview of the different processing steps in this thesis, with a focus on the steps in this chapter: the pattern logic, implementation and adaptation.

3.2 Pattern logic

Section 3.2 describes what codes the projector image incorporates, or in other words what information is encoded in the projected light: the pattern logic. We propose an algorithm with reproducible results (as opposed to e.g. the random approach by Morano et al. [1998]) that generates patterns in the projector image suitable for robotic arm positioning. The next sections will discuss different types of structured light, in order to come to a motivated choice of a certain type, out of which the proposed algorithm follows.

3.2.1 Introduction

Managing the complexity

The inverse of reconstructing a scene is rendering it. Computer graphics studies this problem. Its models become complex as one wants to approach a reality better: a considerable amount of information is encoded in a rendered image. Hence, not surprisingly, the reverse process of decoding a video frame into real world structures, as studied by computer vision, is thus also more difficult than we would like it to be. The complexity sometimes demands simplified models that do not correspond to the physical reality, but are however a useful approximation of it. The pinhole model is an example of such an approximation: it is used as a basis for this thesis, but it discards the existence of a lens in camera

and projector.
Thus, section 3.3 chooses the projector image as simple and clear as possible, not to add more complexity to an already difficult problem. On the contrary: making the decoding step easier is the aim.

The thesis describes a stereo vision system, using a camera and a projector. The most difficult part of stereo vision is reliably determining correspondences between the two images. Epipolar geometry eases this process by limiting the points in one image possibly corresponding to a point in the other image to a line, thus reducing the search space from 2D to 1D. So for each point we need to look for similarities along a line in the other images. Even that 1D problem can be difficult when little or no texture is present: for example visual servoing of a mobile robot along a uniformly coloured wall is not possible using only cameras. For a textured scene finding the correspondences is difficult, and when no texture is present it becomes even impossible. A solution is to replace one of the cameras by a projector and thereby artificially creating texture.

3.2.2 Positioning camera and projector

The camera can be put in eye-to-hand or **eye-in-hand** configuration. In the former case, the camera has a static position observing the end effector, in the latter case the camera is attached to the end effector. Eye-to-hand keeps an overview over the scene, but as it is static, never sees regions of interest in more detail. Eye-in-hand has the **advantage** that a camera moving rigidly with the end effector avoids occlusions and can perceive **more detail** as the robot is approaching the scene. The projected pattern only contains local information and is therefore robust against partial occlusions, possibly due to the robot itself. An extra advantage of the eye-in-hand configuration is that through the encoders we **know** the **position** of the end effector, and through an hand-eye calibration also the position of the camera. This reduces the number of parameters to estimate. Incorporating this extra information makes the depth estimation more reliable.

For these reasons this thesis deals exclusively with a eye-in-hand configuration. Figure 3.3 shows different kinds of eye-in-hand configurations, depending on the position of the projection device(s). Pagès [2005] for example chooses the top rightmost setup, because then for a static scene the projected features remain static, and IBVS can directly be applied. This section will discuss which this thesis chooses and why. This depends on the projector technology: the following section classifies them according to their light source.

Projection technologies

Incandescent light Older models often use halogen lamps. Halogen lamps are incandescent lamps: they contain a heated filament. This technology does not allow the projector to be moved around because the hot lamp filament can break due to vibrations.

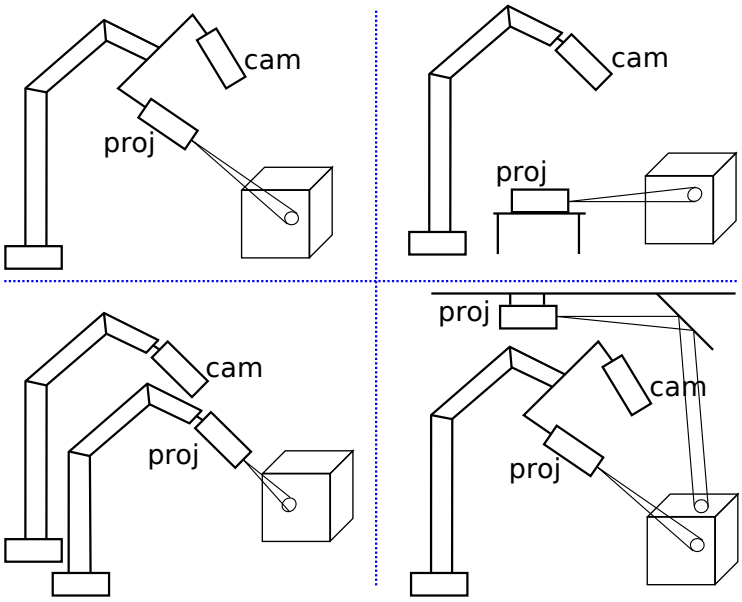


FIGURE 3.3: Different eye-in-hand configurations

Gas discharge light Nowadays most projectors use a gas discharge lamp as these have a larger illuminance output for a constant power consumption. Most projectors on the market at the time of writing use a gas discharge light bulb. Two variants are available that differ in the way the white light is filtered to form the projection image: **L**iquid **C**rystal **D**isplay or **D**igital **L**ight **P**rocessing.

A DLP projector projects colours serially using a colour wheel. These colours usually are the 3 additive primary colours, and sometimes white as a fourth to boost the brightness. A DLP projector contains a **D**igital **M**icromirror **D**evice: a microelectromechanical system that consists of an matrix of small mirrors. The DMD is synchronised with the colour wheel such that the red component for example is displayed on the DMD when the red section of the colour wheel is in front of the lamp. This wheel introduces the extra difficulty of having to adapt the integration time of camera to the frequency of the DLP, otherwise a white projection image may be perceived as having only one or two of its base colours. Concluding: because of its broad availability on the consumer market, and the synchronisation restrictions associated with DLP, this thesis uses an LCD projector, with a gas discharge light bulb.

This gas discharge light bulb is often a metal halide lamp, or – as is the case for the projector used in this thesis – a mercury lamp. These do also have motion restrictions. A mercury lamp for example is even made to be used in a certain position: not only can the projector not move, it has to be put in a horizontal position. As LED and laser projectors are too recent developments, this thesis uses a gas discharge lamp based projector. However, most of the

presented material remains valid for these newer projector types.
Concluding: because of the motion restrictions associated with the lamp, the projector has a static position in the presented experiments, see the top rightmost drawing of figure 3.3. Advantages and disadvantages with respect to the other technologies will become clear in the next sections.

LED light LED projectors have recently entered the market: LEDs are shock resistant and can therefore be moved around. The advantage of their **insensitivity to vibrations** is that one can mount them at the robot end effector together with the camera. The projector can then be the second part of a fixed stereo rig rigidly moving with the end effector. This way, the relative 6D position between camera and projector remains constant, leading to a much simpler calibration.
These projectors are about equally inexpensive as projectors with a gas discharge lamp, but smaller ($< 1000cc$) and lighter ($< 1kg$). They have a much more efficient power use, in the order of 50 lm/W, whereas a projector with a light bulb produces only about 10 lm/W.

Compared to LCD technology that stops the better part ($\pm 90\%$) of the available light, DLP is more thrifty with the available light. So LED projectors are now often used in combination with DLP, which adds the extra synchronisation difficulty again. But also LCD based LED projectors become available.

A disadvantage is their **low illuminance output**: LED projectors currently produce only around $50lumen$. In combination with their high power efficiency this means that they have a much lower power consumption than gas discharge lamp projectors. Projected on a surface of $1m^2$ (a typical surface for this application), this results in an illuminance of $50lux$, about the brightness of a family living room. Therefore, under normal light conditions this technology is (still) inadequate: the contrast between ambient lighting and projected features is insufficient. However, in applications where one can control the lighting conditions, and the whole robot setup can be made dark, LED projectors can be used. Thus under these conditions, one can attach not only the camera but also the projector to the robot end effector, according to the top leftmost drawing of figure 3.3. This removes the need to adapt the calibration during robot motion, and thus makes the calculations mathematically simpler and less prone to errors. Moreover, self occlusion is much less likely to occur. In this work however, we do not assume a dark environment and work with a gas discharge lamp projector. Within the available gas discharge LCD projectors, the chosen projector is one that can focus a small image nearby in order to have a finer spatial resolution.

Laser light The Fraunhofer institute [Scholles et al., 2007] recently proposed laser projectors using a micro scanning mirror. The difference with a DMD is that a DMD is an array of micromirrors, each controlling a part of the image (spatial multiplexing). This micro scanning mirror on the other hand is only one mirror that moves at a higher frequency (temporal multiplexing), producing a Lissajous figure. The frequency of the two axes of the mirror are chosen such

that the laser beam hits every virtual pixel within a certain time, defined by the frame rate. By synchronising laser and mirror any monochrome image can be projected. If one combines a red, green and blue laser, a coloured image is also possible (white light can also be produced). An advantage of this technology for robot arm applications, is the physical **decoupling** between the light source and the image formation. They are linked using a flexible optical fiber: the light is redirected from a position that is fixed with respect to the world frame to a position that is rigidly attached to the end effector frame. In this case, there is no need any more for an expensive and complex multipixel fiber, as would be the case without this decoupling.
The laser can remain static, while the projection head is moving rigidly with the end effector. A static transformation from the projector frame to the camera frame makes the geometrical calibration considerably easier, just like the LED projectors. However, low light source power is less of a problem here. Figure 3.4 demonstrates this setup. Transmitting the light over fiber wire is optically more difficult using a gas discharge projector.

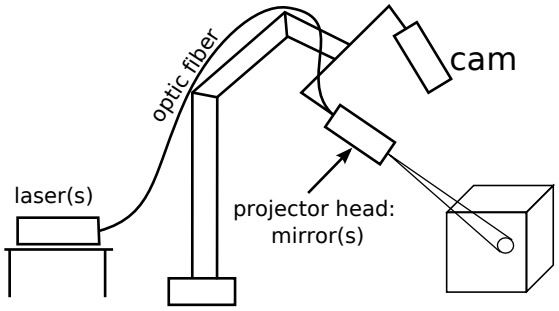


FIGURE 3.4: A robot arm using a laser projector

The combination of a fiber optic coupled laser with a DMD seems a better choice for sparse 3D reconstruction, as it can inherently project isolated features. With a micro scanning mirror, the laser has to be turned on and off for every projection blob. It requires much higher frequencies and better synchronisation to obtain the same result. To our knowledge, DMD operation under laser illumination has not been studied thoroughly yet. This thesis does not study this technology as it is quite recent, but it seems to be a promising research path, especially for endoscopy (see the experiments chapter).

Other projector configurations

There are other projection possibilities. Consider for example the setup on the bottom left side of figure 3.3: a **projector and** a **camera moving independently** on two robot arms. Clearly, this projector would have to be of the LED or laser type to be able to move. The advantage of this setup is that both imaging devices are independent and the arm with the projector can thus be

constrained to assume a mathematically ideal (conditioning) position with respect to the camera. However, the calibration tracking is complex in this case, as both devices move independently. If the projector is able to move, the setup on the top left of figure 3.3 – where they are rigidly attached to each other – is more attractive. The latter leads to simpler, and thus often more robust, mathematics.

One could also use **multiple projectors**, for example to estimate the depth from different viewpoints, like in [Griesser et al., 2006]. Different configurations of moving or static projectors are possible. Consider the setup on the bottom right side of figure 3.3: a fixed projector can for example be using a mirror to ensure that a sparse depth estimation of the scene is always available to the robot, and not occluded. The projection device attached to the robot arm, for example a laser projector, can project finer, more local patterns to actively sense details in a certain part of the camera image, that are needed to complete the robot task. Clearly, one needs to choose different types of projection patterns for the different projectors to be able to discern which projected feature originated from which projector.

1D versus 2D encoding

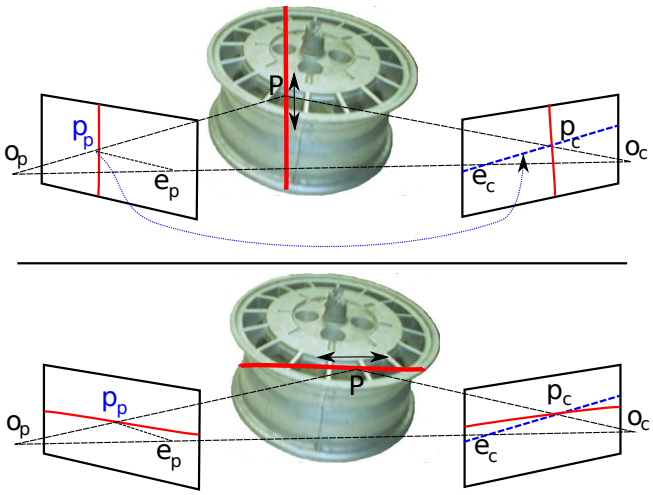


Figure 3.5: Good conditioning of line intersection in the camera image (top) and bad conditioning (down)

In order to estimate the depth, each pixel in the camera and projector image is associated with a ray in 3D space. Section 4.3.1 descibes how to estimate the opening angles needed for this association. The crossing (near intersection) between the rays define the top of the triangle. 1D structured light patterns exploit the epipolar constraint: they project only vertical lines when the camera is positioned beside the projector, see figure 3.5. In that case, the intersection of

the epipolar lines $e_c\bar{p}_c$ in the camera image (corresponding to p_p in the projector image) and the projection planes (stripes) is conditioned much better than for horizontally projected lines. Analogously in a setup where the projector is on top or below the camera: there horizontal lines would be best. Salvi et al. [2004] present an overview of projection techniques, among others using such vertical stripes.

Before calibration there is no information on the relative rotation between camera and projector. For robustness reasons, we plan on self calibrating the setup, see section 4.4. Therefore we designed a method to generate **2D patterns** such that the correspondences are independent of the unknown relative orientation. In addition larger minimal Hamming distances between the projected codes provide the pattern with error correcting capabilities, see section 3.2.4. All correspondences can be extracted from one image, the pattern is suitable for tracking a moving object.

Conclusion

At the beginning of this work, LED and laser projectors were not yet available. Therefore, this thesis mainly studies the use of the established, consumer market technology with a gas discharge lamp. The pose between camera and projector are therefore variable, and the use of 2D projector patterns the easiest method (see figure 1.1).

3.2.3 Choosing a coding strategy

3.2.3.1 Temporal encoding

Temporal encoding uses a time sequence of patterns, typically binary projection patterns.
In order to be able to use the epipolar constraints, we need to calibrate the setup: estimating the intrinsic and extrinsic parameters. 2D correspondences between the images are necessary to calculate the extrinsic parameters. So, during a calibration phase typically both horizontal and vertical patterns are projected to retrieve the 2D correspondences. After calibration on the other hand, only stripes in one direction are projected, as one can then rely on epipolar geometry. Before calibration, the rotation between camera and projector in the robot setup is unknown. For a static scene temporal encoding can find the correspondences to help retrieve this rotation. A time sequence of binary patterns is projected in two directions onto a calibration grid.

Salvi et al. [2004] summarise how associating the pixels in camera and projector image can be done using time multiplexing (this section) or only in the image space (section 3.2.3.2). In the time based approaches the scene has to remain static as multiple images are needed to define the correspondences. In the eighties binary patterns were used. In the nineties these were replaced by Gray code patterns [Inokuchi et al., 1984], the advantage being that consecutive codewords then have a Hamming distance of one: this increases robustness

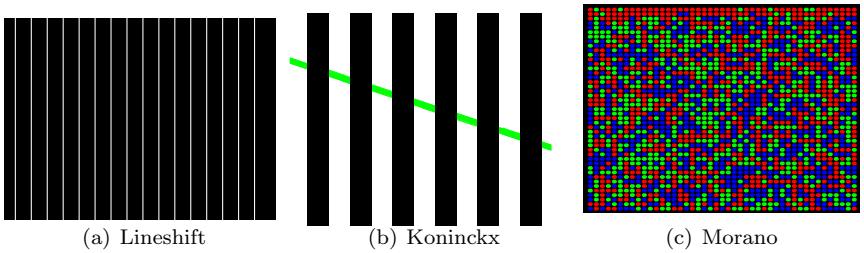
to noise. Phase shifting (using sine patterns) is a technique that improves the resolution. Gühring [2000] observes that phase shifting has some problems. For example, when a phase shifted pattern is projected onto a scene, phase recovering has systematic errors when the surface contains sharp changes from black to white. He proposes line shifting instead, illuminating one projector line at a time, see figure 3.6 on the left. Experiments with flat surfaces show that phase shifting recovers depth differences of $\pm 1mm$ out of the plane (thus differences that should be 0) caused by different reflectance properties on a flat surface. Line shifting substantially reduces this unreal depth difference. Another advantage is that this system has less problems with optical crosstalk (the integration of intensities over adjacent pixels). A disadvantage is that with this system the scene has to remain static during 32 different projected patterns: the price to pay is more projected patterns.

The previous methods require offline computation, more recent methods perform online range scanning. Hall-Holt and Rusinkiewicz [2001] propose a system of only 4 different projection patterns in black and white. The two boundaries of each stripe encode 2 bits, resulting in a codeword of one byte after 4 decoded frames. Their system defines 111 vertical stripes, so 2^8 possibilities are more than sufficient to encode it. This time-based system does allow for some movement of the scene, because the stripe boundaries are tracked. However, this movement is limited to scene parts moving half a stripe per decoded frame. This corresponds to a movement of ± 10 percent of the working volume a second (for a scene at $\pm 1m$).
Vieira et al. [2005] present a similar online technique: also a 1D colour code that also needs 4 frames to be decoded. For a white surface this code would only be 2 frames long. However, coloured surfaces do not reflect all colours of light (see section 3.3.2). Therefore after each frame, the in colour complementary frame is also projected. An advantage of this technique compared to the work of Hall-Holt and Rusinkiewicz [2001] is that it can also retrieve texture information: it can reconstruct the scene colours without the need to capture a frame with only ambient lighting. This is potentially useful if one needs to execute 2D vision algorithms, apart from the 3D reconstruction, to extract more information. Section 8.4 elaborates on the combination of 2D and 3D vision.

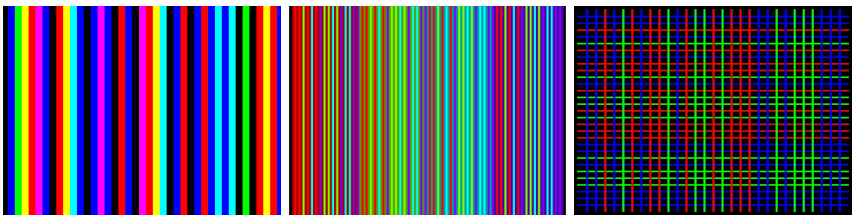
3.2.3.2 Spatial encoding

One-shot techniques solve the issue of the previous paragraph that the scene has to remain static during several image frames. This work also studies moving scenes: it needs a method based on a single image. As several images contain more information than one, there is a price to pay: in the resolution. Hence one exchanges resolution for speed.

Koninckx et al. [2003] propose a single shot method using vertical black and white stripes, crossed by one or more coloured stripes, see figure 3.6. By default this stripe is green, since industrial cameras often use a Bayer filter, making them more sensitive to green. If the scene is such that the green cannot be detected, another colour is used automatically. The intersection between coloured stripes



(a) Lineshift (b) Koninckx (c) Morano



(d) Zhang (e) Pagès (f) Salvi

FIGURE 3.6: Projection patterns

and the vertical ones, and the intersection with the – typically horizontal – epipolar lines both need to be as well conditioned as possible. The angle at which these stripes are, is therefore a compromise between these two. Moreover, the codification of this system is sparse, as only the intersection points between stripes and coding-lines are actually encoded.

There are other single shot techniques that do not suffer from this conditioning problem, in which the pattern does not consist of lines, but of more compact projective elements. Each projected feature, often a circular blob, then corresponds to a unique code from an alphabet that has at least as many elements as features that need to be recognised. Morano et al. [1998] points out that one can separate the letter in the alphabet of each element of the matrix, and the representation of that letter. Just like this thesis separates the sections *pattern logic* (3.2) and *pattern implementation* (3.3).

In algebraic terms, the projective elements are like letters of an alphabet. Let a be the size of the alphabet. The encoding can then for example be in a different colours (see the pattern of Morano in figure 3.6), intensities, shapes, or a combination of these elements. Projecting an evenly spread matrix of features with m rows and n columns requires an alphabet of size $n\ m$. Consider a 640×480 camera and assume the best case scenario that the whole projected scene is just visible). Then projecting a feature every 10 pixels requires a 64×48 grid and thus more than 3000 identifiable elements. Trying to achieve this using for example a different colour for each feature – or intensity in the grey scale case – is not realistic as the system would be too sensitive to noise: this method is called *direct coding*.

One could imagine projecting more complex features than these spots: sev-

eral spots next to each other form one projective element together. This way, we need considerably less different intensities to represent the same number of possibilities: it drastically reduces the size of the alphabet needed. However, the projected features then become larger, which increases the probability that part of it is not visible, due to depth discontinuities for example. A solution is to also define a feature as a collection of spots, but sharing spots with adjacent features. Section 3.2.3.3 explains this concept further.

3.2.3.3 Spatial neighbourhood

The number of identifiable features required can be reduced by taking into account neighbouring features of each feature. This is called using spatial neighbourhood. An overview of work in this field can be found in [Salvi et al., 2004]. In this way, we reduce the number of identifiable elements needed for a constant amount of codes.
In terms of information theory, this spatial neighbourhood uses the communication channel digitally (a small alphabet). Direct coding on the other hand is the near analogue use of the channel: the number of letters in the alphabet is large, only limited by the colour discretisation and the resolution of the projector. Let $\mathcal{W}_p$ be the width in pixels of the projector image, and $\mathcal{H}_p$ the height. If the projector is driven in RGB using one byte/pixel/channel, the maximum size of the alphabet is $min(255^3, \mathcal{W}_p\mathcal{H}_p)$ (there cannot be more letters in the alphabet than there are pixels in the image).

1D patterns To make the elements of a 1D pattern uniquely identifiable, a De Bruijn sequence can be used. That is a cyclic sequence from a given alphabet (size a) for which every possible subsequence of a certain length is present exactly once, see [Zhang et al., 2002] for example. The length of this subsequence is called the window w. These subsequences can be overlapping. Of course, using 1D patterns requires a calibrated setup to calculate the epipolar geometry. A 2D pattern – or two perpendicular 1D patterns – is necessary to calibrate it.

The work by Zhang et al. [2002] is an example of a stripe pattern: every line – black or coloured – represents a code, see figure 3.6. Its counterpart is multi-slit patterns, where black gaps are left in between the slits (stripes). The advantage of stripe patterns over multi-slit patterns is that the resolution is higher, since it doesn't need space for the black gaps. However more different projective elements are required since adjacent stripes must be different, and the more elements to be distinguished the more chance that noise will induce a erroneous decoding.

Pagès et al. [2005] combines the best of both worlds in a 1D pattern based on De Bruijn sequences, also represented in figure 3.6. In RGB space this pattern is a stripe pattern, but converted to grey scale, darker and brighter regions alternate as a multi-slit pattern. So both edge and peak based algorithms segment the images, and Pagès et al. thus increases the resolution here while the alphabet size remains constant (using 4 hue values).

2D patterns

Grid patterns A 2D pattern is not only useful for the initial calibration of the setup of figure 1.1 but also online, since a constantly changing baseline requires the extrinsic parameters to be adapted constantly, section 4.4 explains this calibration. If 1D patterns would be used, the conditioning of the intersection between the epipolar lines and the projected lines can become infinitely bad during the motion. With 2D grid patterns it never becomes worse than the conditioning of the intersection of lines under 45°.

Salvi et al. [1998] extends the use of De Bruijn sequences to 2D patterns. He proposes a 2D multi-slit pattern that uses 3 letters for horizontal, and 3 for vertical lines. These letters are represented by different colours in this case, see figure 3.6. To maximise the distance between the hue values, the 3 additive primaries (**r**ed, **g**reen and **b**lue) are used for one direction, and the 3 subtractive primaries (**c**yan, **y**ellow and **m**agenta) for the other. Both directions use the same De Bruijn sequence, with a window property of $w = 3$. The corresponding segmentation uses skeletonisation and Hough transform. The Hough transform in itself is rather robust: that is, the discretisation and the transformation are. But the last step in the process, the thresholding, is not. The results are very sensible to the chosen thresholds.
Furthermore, this assumes that the objects of the scene are composed of planar parts: it is problematic for a strongly curved scene. As only the intersection of the grid lines are encoded, one could try to avoid a Hough transformation and the dependency on a planar scene, by attempting to somehow detect crosses in the camera image. For an arbitrarily curved scene, the robustness of this segmentation seems questionable. In addition, this technique does not allow to work with local relative intensity differences, as explained in section 3.3.7, and would have to rely on absolute intensity values. As a solution, dots can replace these lines as projective elements. For these reasons, this thesis projects a matrix of compact elements that are not in contact with one another.

Matrix of dots The section about the pattern implementation, section 3.3, will explain why filled circles are the best choice for these compact elements, better than other shapes. A 2D pattern is used for both the geometric (6D) calibration and during online reconstruction, since during the latter phase one needs to adapt the calibration. The section about calibration tracking, section 4.5, explains why only feedforward prediction of the next 6D calibration is insufficient. For this feedforward a 1D pattern would suffice, as no visual input is needed. The system needs a correction step: for an uncalibrated setup (the previous calibration has become invalid), one cannot rely on epipolar geometry and hence a 2D pattern is necessary during this step.

For a pattern of dots in a matrix, the theory of perfect maps can be used: **a perfect map is the extension of a De Bruijn sequence in 2D**. It has the same property as De Bruijn sequences, but in 2D: for a certain rectangular submatrix and rectangular matrix size, every submatrix occurs only once

in a matrix of elements from a given alphabet. These perfect maps can be constructed in several ways. Etzion [1988] for example presents an **analytical** construction algorithm. He uses De Bruijn sequences: the first column is a De Bruijn sequence, and the next columns consist of cyclic permutations of that sequence, each column shifted one position in the sequence. A disadvantage of this technique is that there is a fixed relation between the size of the desired submatrix $w \times w$ and the size of the entire matrix $r \times c$, namely $r = c = a^w$, where a^w is the length of the corresponding De Bruijn sequence.

Another interesting algorithm to generate a perfect map pattern is the one by Chen et al. [2007]. Most of that algorithm is also analytical (the first step contains a random search, but only in one dimension). It is built up starting from a 1D sequence with window property 3. But since this pattern is fully illuminated (no gap in between the projected features), no neighbouring spots can have the same code (colour) in this case. Therefore the 1D sequence that forms the (horizontal) basis of the perfect map is only of length $a(a-1)+2$ instead of a^3 for a De Bruijn sequence. Chen et al. [2007] combine this sequence with another 1D sequence to generate a 2D pattern analytically, and thus efficiently. The technique is based on 4-connectivity (north, south, west and east neighbours) instead of 8-connectivity (including the diagonals). This means less degrees of freedom in generating the patterns, or in other words, more letters in the alphabet (different colours) needed to achieve a pattern of a certain size. Indeed, the patterns are of size $[(a-1)(a-2)+2] \times [a(a-1)^2+2]$. For example, a 4 colour set generates a 8×38 matrix and a 5 colour set a 14×82 matrix. These matrices are elongated, which is not very practical to project on a screen with 4:3 aspect ratio.

3.2.4 Redundant encoding

Error correction

2 main sources of errors need to be taken into account:

- Low level vision: decoding of the representation (e.g. intensities) can be erroneous when the segmentation is unable to separate the projected features. Hence, it makes mistakes in the **data association** between camera and projector image features.
- Scene 3D geometry: **depth discontinuities** can cause occlusions of part of the pattern. As explained in Morano et al. [1998] the proposed pattern is able to deal with these discontinuities because every element is encoded w^2 times: once for every submatrix it belong to. A voting strategy is used here: for every element the number of correct positives are compared to the number of false positives. Robustness can be increased by increasing this signal to noise ratio: make the difference in code of every submatrix compared to the codes of every other submatrix, larger.

In order to be able to correct n false decodings in a code, the minimal Hamming distance h between any of the possible codes has to be at least $2n+1$. Only

requiring each submatrix to be different from every other submatrix produces a perfect map with $h = 1$, and hence no error correction capability. Requiring for example 3 elements of every submatrix to be different from every other submatrix, makes the correction of one erroneous element possible. Or, to put it in voting terms: every element in a submatrix could be wrong, discarding each one of the w^2 elements of the code at a time labels all elements of that submatrix w^2 times. As each element is part of w^2 submatrices, the number of times an element is labelled (also called the confidence number, comparable with the signal of the S/N ratio) can be as high as w^4.

Choosing the desired window size w

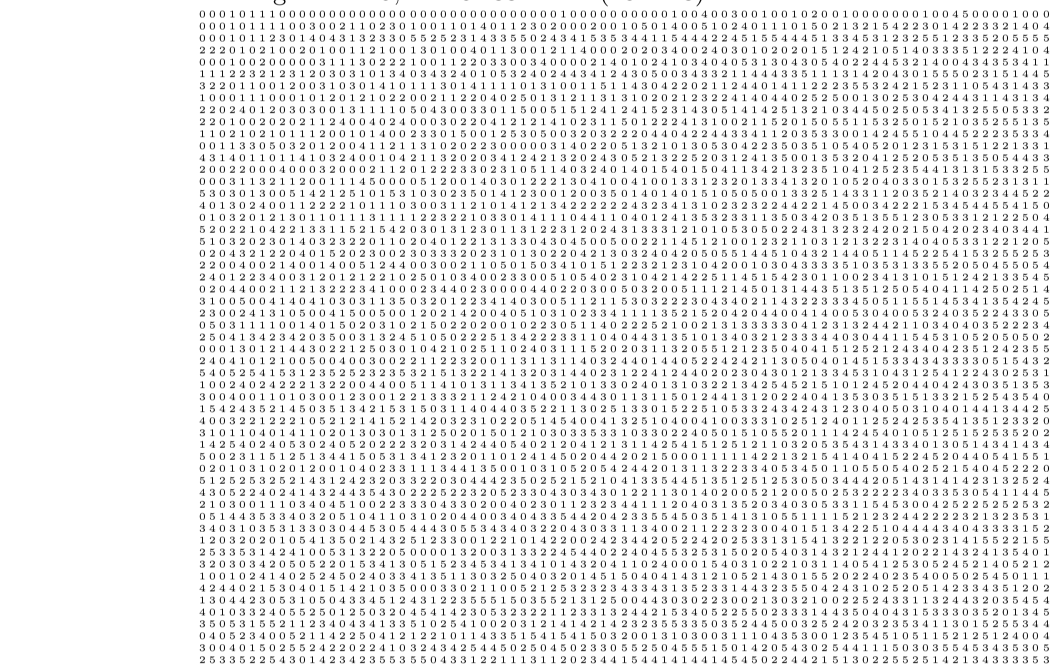
The larger w, the less local every code is and the more potential problems there are with depth discontinuities. Therefore it is best to choose w as low as possible. An even w is not useful, because in that case no element is located in the middle. $w = 1$ is direct encoding: not a realistic strategy. $w = 5$ means that in order to decode one element, no less than 25 elements must be decodable: the probability of depth discontinuity problems becomes large. In addition, it is overkill: in order to find a suitable code for sparse reconstruction, one does not need this amount of degrees of freedom: $w = 3$ suffices. Therefore we choose $w = 3$.

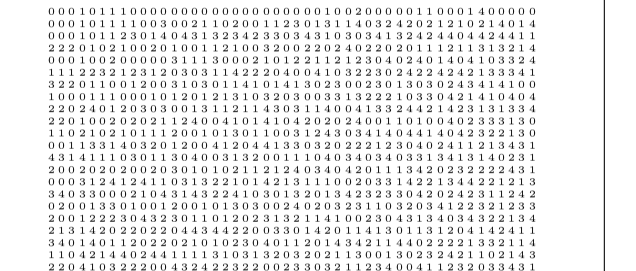
Choosing the desired minimal Hamming distance h

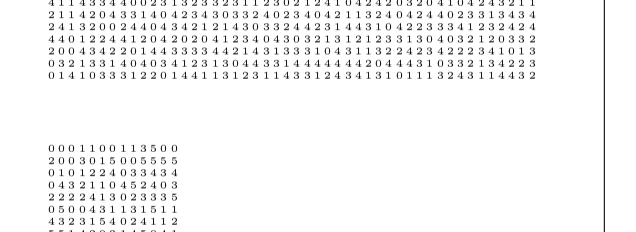
The larger h the better during the decoding process, but the more difficult the pattern generation: two submatrices will more often not be different enough. Or in other words, the larger h, the more restrictive the comparison between every two submatrices. The projected spots should not be too small as they cannot be robustly decoded anymore then. Assume that one is able to decode an element every 10 pixels (which is rather good, as every blob is then as small as 6×6 pixels with 4 pixels black space in between). For a camera with VGA resolution, this means we need a perfect map of about 48×64 elements. Requiring $h = 3$ in our technique quickly yields a satisfactory 64×85 result for $a = 6$, or 36×48 for $a = 5$. Hence in the experiments, we choose $a = 5$ as it is a suitable value to produce a perfect map that is sufficiently large for this application. With $h = 5$ the algorithm does not find a larger solution than 10×13. To increase the size of this map, one would have to choose a larger a. With $h = 1$, the above algorithm quickly produces a 398×530 perfect map for $a = 5$: needlessly oversized for our application: one can choose any submatrix of appropriate size. Figure 3.7 illustrates this result of our algorithm.

Pagès et al. [2006] use the same setup of a fixed projector and a camera attached to the end effector but do not consider what happens when the view of the pattern is rotated over more than $\frac{\pi}{4}$. There a standard Morano pattern of 20×20, $h = 1$, $a = 3$ is used. As that pattern is not rotationally invariant, rotating over more than $\frac{\pi}{4}$ will lead to erroneous decoding, as some of the rotated features have a code that is identical to non rotated features in other locations in the camera image.

FIGURE 3.7: Result for $w = 3$: on the left for $a = 6$, $h = 3$: 5146 codes (64×85); top right: $a = 5$, $h = 3$: 1564 codes (36×48), bottom right: $a = 6$, $h = 5$: 88 codes (10×13)







3.2.5 Pattern generation algorithm

Pattern rotation

In the setup used throughout this thesis, see figure 1.1, the rotation between camera and projector image can be arbitrary. Thus, starting from an uncalibrated system, the rotation is unknown. Therefore, during calibration (see 4.4) each submatrix of the projected pattern can occur only once using the same orientation, but also only once rotating the pattern over an arbitrary angle. Perfect maps imply an organisation of projected entities in the form a a matrix: all elements are at right angles. Hence only rotating over $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$ covers all possible orientations. This thesis calls that property **rotational invariance**. "Formula" 3.1 illustrates this property: let $c_{i,j}$ be the code element at row i and column j and $\forall i,j : c_{i,j} \in \{0,1,\ldots,a-1\}$, then all 4 submatrices represent the same code:

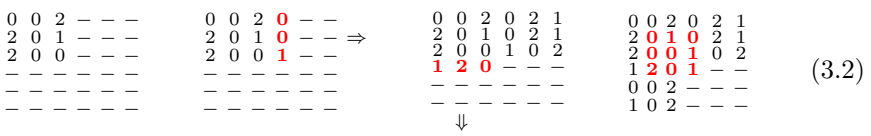
$$\begin{array}{cccc} c_{0,0}\,c_{0,1}\,c_{0,2} & c_{0,2}\,c_{1,2}\,c_{2,2} & c_{2,2}\,c_{2,1}\,c_{2,0} & c_{2,0}\,c_{1,0}\,c_{0,0} \\ c_{1,0}\,c_{1,1}\,c_{1,2} & c_{0,1}\,c_{1,1}\,c_{2,1} & c_{1,2}\,c_{1,1}\,c_{1,0} & c_{2,1}\,c_{1,1}\,c_{0,1} \\ c_{2,0}\,c_{2,1}\,c_{2,2} & c_{0,0}\,c_{1,0}\,c_{2,0} & c_{0,2}\,c_{0,1}\,c_{0,0} & c_{2,2}\,c_{1,2}\,c_{0,2} \end{array} \tag{3.1}$$

None of the analytic construction methods provide a way to generate such patterns. The construction as proposed by Etzion [1988] for example is not rotationally invariant. This can be proved by rotating each $w \times w$ submatrix $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$ and then comparing it to all other – unrotated – submatrices: the same submatrices are found several times. The pattern proposed by Chen et al. [2007] is also not rotationally invariant.

So, when the rotation is unknown perfect maps need to be constructed in a different way. One could try simply testing all possible matrices. However, the computational cost of that is prohibitively high: a^{c*r} matrices have to be tested, for example for the rather modest case of an alphabet of only 3 letters (e.g. using colours: red, green, blue) and a 20×20 matrix, this yields $3^{400} \approx 10^{191}$ matrices to be tested.

Algorithm design

Morano et al. [1998] proposed another brute force method, but a more efficient one. For example, the construction of a perfect map with $w = 3$, $a = 3$, for a 6×6 matrix, is according to this diagram, which will now be clarified:



(3.2)

First the top left $w \times w$ submatrix is randomly filled: on the left in diagram 3.2 above. Then all $w \times 1$ columns right of it are constructed such that the

uniqueness property remains valid: they are randomly changed until a valid combination is found (second part from the left of diagram 3.2). Then all $1 \times w$ rows beneath the top left submatrix are filled in the same way (third drawing). Afterwards every single new element determines a new code: the remaining elements of the matrix are randomly chosen always ensuring the uniqueness, see the rightmost part of diagram 3.2. If no solution can be found at a certain point (all a letters of the alphabet are exhausted), the algorithm is aborted and started over again with a different top left submatrix.
In this way Morano can cope with any combination of w, r and c, but the algorithm is not meant to be rotationally invariant. We propose a new algorithm, based on the Morano algorithm, but altered in several ways:

- **Adding rotational invariance.** Perfect maps imply square projected entities, so out of every spot with its 8 neighbours, 4 are closer and the 4 others are a factor $\sqrt{2}$ further. This means that the only extra restrictions that have to be satisfied in order for the map to be rotationally invariant, are rotations over $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$. Thus, while constructing the matrix, each element is compared using only 4 rotations. Each feature is now less likely to be accepted than in the Morano case. However having only 3 extra constraints to cover all possible rotations, keeps the search space of all possible codes relatively small, increasing the chances of finding a valid perfect map.
- The matrix is constructed without first determining the first w rows and first w columns. In this way, larger matrices can be created from smaller ones without the unnecessary constraints of these first rows and columns. There is no need for specifying the final number of columns or rows at the beginning of the algorithm. Hence we solve the problem **recursively**: at each increase in matrix size, w new elements are added to start a new column, and w to start a new row, in the next steps the matrix can be completed by adding one element at each time.
- At each step, a certain subset of elements needs to be chosen: the $w \times w$ elements of the top left submatrix in the beginning, the w elements when a new row or column is started, or 1 element otherwise. Putting each of these matrix elements after one another yields a huge base a number. In the algorithm the pattern is augmented such that this number always increases, so **a perfect map candidate is never checked twice**. We assume a **depth-first** search strategy: at each iteration the elements that can be changed at that point (size w^2, w or 1) are increased by 1 base a, until the $w \times w$ submatrix occurs only once (considering the rotations). When increasing by 1 is no longer possible, we assign 0 to the changeable elements and use backtracking to increase the previous elements. First the elements that violate the constraints are changed, only if that is not possible, we alter other previous elements. In this way only promising branches of the tree are searched until the leaves, and restarting from scratch is never needed.

$$\begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \\ c_{4,0} & c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} \\ c_{5,0} & c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & c_{5,5} \end{bmatrix} \Rightarrow \begin{array}{l} c_{0,0}c_{0,1}c_{0,2}c_{1,0}c_{1,1}c_{1,2}c_{2,0}c_{2,1} \\ c_{2,2}c_{0,3}c_{1,3}c_{2,3}c_{3,0}c_{3,1}c_{3,2}c_{3,3} \\ c_{0,4}c_{1,4}c_{2,4}c_{4,0}c_{4,1}c_{4,2}c_{3,4} \\ c_{4,3}\,c_{4,4}\,c_{0,5}c_{1,5}c_{2,5}c_{5,0}c_{5,1} \\ c_{5,2}c_{3,5}c_{4,5}c_{5,3}c_{5,4}c_{5,5} \end{array} \tag{3.3}$$

The size to the search space is only a ath of the space used by Morano, as the absolute value of each element of the matrix is irrelevant. Indeed, the matrix is only defined up to a constant term base a, as the choice of which letter corresponds to which representation (e.g. colour) is arbitrary. So we can assume the top left element of the matrix to be zero. Agreed that the remaining search space remains huge, therefore the described search strategy is necessary.

- The pattern need not be square, one can specify **any aspect ratio**. For example for an XGA (1024×768) projector, the aspect ratio is $4:3$: for every fourth new column no new row is added.

After the calibration one could use a pattern without the extra constraint of the rotational invariance, and rotate this pattern according to the camera movement. But this implies more calculations online, which should be avoided as they are a system bottleneck. Moreover, rotating the pattern in the projector image would complicate the pose estimation, even when the scene is static. Indeed, moving the projector image features because the camera has rotated, implies that the reconstruction points change. Keeping the reconstructed points the same over time for a static scene, facilitates pose estimation and later object recognition.

Another possibility is to keep the rotation in the projector pattern constant, but constantly calculate what is up and what is down in the camera image. This would again imply unnecessary online calculations. Moreover, the results (section 3.2.7) show that the rotational invariance on average require only one extra letter to reach the same matrix size with a constant minimal Hamming distance. Hence, we choose to use the rotational invariant pattern both for calibration and online reconstruction and tracking. Since the estimation of the orientation online is also a good option, section 3.2.7 also presents the results without the rotational invariance constraint.

Algorithm outline

The previous section presented the requirements for the algorithm, and changes compared to known algorithms. This section presents an outline of the resulting algorithm, where all these requirements have been compiled. Note that this is only a description for easy understanding of the important parts: assume $w = 3$ for easier understanding. Appendix A describes the pattern construction algorithm in detail.

- For every step where matrix elements can be corrected, remember which of the 9 elements can be changed in that step, and which were already determined by previous steps. This corresponds to the method *calcChangable* in appendix. For example:
  - index 0: mark all elements of the upper left 3×3 submatrix as changable.
  - index 1: add a column: mark elements $(0, 3)$ through $(2, 3)$.
  - index 2: is the aspect ratio times the number of columns large enough to add a row? In this case $round(4 \cdot 3/4)$ is not larger than 3, so first another column is added: mark elements $(0, 4)$ through $(2, 4)$ as changable.
  - index 3: since $round(5 \cdot 3/4)$ is larger than 3, a row is added: first mark $(3, 0)$ through $(3, 2)$ as changable.
  - then mark $(3, 3)$ for index 4, and $(3, 4)$ for index 5 ...
- For ever larger matrices:
  - For the current submatrix, say s, check whether all previous submatrices are different, according to a given minimal Hamming distance, and rotating the current submatrix over 0, 90, 180 and 270° at each comparison.
  - If it is unique, move to the next submatrix/index of the first step.
  - If not, convert the changable elements of that submatrix in a base a string. Increment the string, and put the results back into the submatrix.
  - If this string is already at its maximum, set the changable elements of this submatrix to 0, and do the same with all changable elements of the previous submatrices, up to the point where the submatrix before the one just reset, has changable elements that are part of the submatrix s. Increment those elements if possible (if they are not at their maximum), otherwise repeat this backtracking procedure until incrementation is possible or all 9 elements of s have been reset.
  - If all 9 elements of s have been reset, and still no unique solution has been found, reset all previous steps up to the step that causes the conflict with submatrix s, and increment the conflicting submatrix. If this incrementation is impossible, reset all changable elements of the previous steps until other elements of the conflicting submatrix can be increased, as before.
  - If this incrementation is not possible, reset the conflicting element, and backtrack further: increasing elements and resetting where necessary.
  - If the processing returns to the upper left submatrix, the search space is exhausted, and no larger pattern can be found with the given parameters.

Complexity
The algorithm searches the entire space, and therefore always finds the desired pattern if there is a pattern in the space that complies to the desired size, window size and Hamming distance. It does not garantee that this pattern will under all circumstances be found within reasonable time limits. However, the results (see section 3.2.7) show that it does find all patterns that may be necessary in this context within a small amount of time, and in a **reproducible** way: the main merit of the algorithm is a smart **ordering** of the search in the pattern space. Clearly, if there is no pattern in the search space that complies to the demands, or the computing time is unacceptable, one can, if the application permits it, weaken the restrictions by increasing the alphabet size or decreasing the required Hamming distance.

It would be interesting to be able to produce these patterns analytically. Constructing general perfect maps belongs to the EXPTIME complexity class: no algorithm is known to construct them in polynomial time, testing all the possibilities requires a^{rc} steps. Etzion [1988] proposes an analytical algorithm, but not for the general case: the patterns are square, of Hamming distance 1 and not rotationally invariant.

3.2.6 Hexagonal maps

Adán et al. [2004] use a hexagonal pattern instead of a matrix. Each submatrix consists of 7 elements: a hexagon and its central point. Starting from a matrix form, this can be achieved by shifting the elements in the odd columns half a position down and making these columns one element shorter than their counterparts in the even columns.

An advantage of hexagonal maps is that the distance to all neighbours is equal. If precision is needed, this distance can be chosen as small as the smallest distance which is still robustly detectable for the low level vision. In the perfect map case, the corner (diagonal) elements of the squares are further away than the elements left, right, above and below the centre. In other words, in the matrix organisation structure there are elements that are further apart than minimally necessary for the segmentation, which is not the case for the hexagonal structure. Hence, the chance of failure due to occlusion of part of the pattern is minimal in the hexagonal case, but not in the matrix organisation case.

Also, the total surface used in the projector image can be made smaller than in the matrix organisation case for a constant number of projected features. Say the distance between each row is d (the smallest distance permitted by low level vision) then the distance between each column is not d as in the matrix structure case, but $\sqrt{3}(d/2)$, permitting a total surface reduction of 86.6%. These are two – rather small – advantages concerning accuracy.
Adán et al. [2004] use colours to implement the pattern and encode every submatrix to be different if its number of elements in each of the colours is different: the elements are regarded as a **set**, their relative positions are discarded. The advantage of this is that it slightly reduces the online computational cost. Indeed, one does not have to reconstruct the orientation of each submatrix. His

hexagonal pattern is therefore also rotationally invariant, a desired property in the context of this thesis with a moving camera. The number of possible codes is a (for the central element) times the combination with repetition to choose 6 elements out of a : $a\binom{a+6-1}{6}$. For example, for $a = 6$ the number of combinations is 2772, which is a rather small number since codes chosen from this set have to fit together in a matrix. Hence, Adán et al. use a slightly larger alphabet with $a = 7$, resulting in 6468 possibilities.
Restricting the code to a set ensures rotational invariance, and avoids adding an online computational cost. However, there are less stringent ways, other than restricting the code to a set, to achieve that result. It is sufficient to consider all codes up to a cyclic permutation. This is less restrictive while constructing the matrix and should allow the construction of larger matrices and/or matrices with a larger minimal Hamming distance between its codes. Since the code length $l = 7$ is prime, the number of possible cyclic permutations for every code is l. Except in a cases, when all elements of the code are the same, then no cyclic permutation exist. All cyclic permutations represent the same code. Therefore, the number of possible codes is $a + \frac{a^l - a}{l}$, equal to 39996 for $a = 6$ or 117655 for $a = 7$, considerably larger than respectively 2772 and 6468. Hence, this drastically increases the probability of finding a suitable hexagonal map.

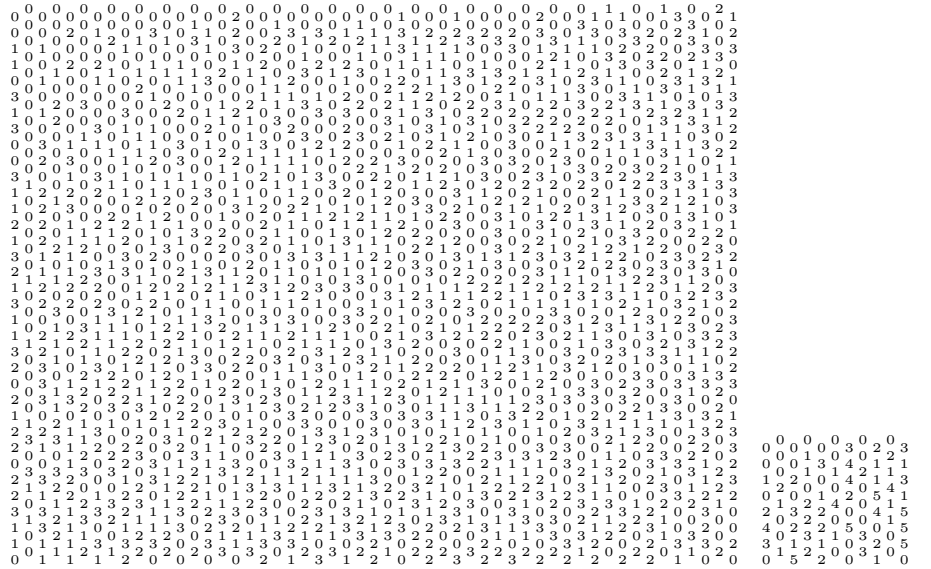


FIGURE 3.8: Left: result with $w = 3$, $a = 4$, $h = 1$: 1683 codes (35×53); right: $w = 3$, $a = 6$, $h = 3$: 54 codes (8×11)

We implemented an algorithm for the hexagonal case, similar to the one for the matrix structure. 6 possible rotations have to be checked for every spot instead of 4. This, in combination with a smaller neighbourhood (6 neighbours

TABLE 3.1: Code and pattern sizes for rotationally independent patterns

$a \backslash h$	1	3	5
2	108 (11×14)	6 ($4 \times 5^*$)	2 ($3 \times 4^*$)
3	3763 (55×73)	63 (9×11)	6 ($4 \times 5^*$)
4	51156 (198×263)	352 (18×24)	20 (6×7)
5	209088 (398×530)	1564 (36×48)	48 (8×10)
6	278770 (459×612)	5146 (64×85)	88 (10×13)
7	605926 (676×901)	15052 (108×144)	165 (13×17)
8	638716 (694×925)	35534 (165×220)	192 (14×18)

TABLE 3.2: Code and pattern sizes for patterns without rotational independence constraint

$a \backslash h$	1	3	5
2	391 (19×25)	24 (6×8)	6 ($4 \times 5^*$)
3	14456 (106×141)	192 (14×18)	24 (6×8)
4	131566 (316×421)	1302 (33×44)	63 (9×11)
5	243390 (429×572)	5808 (68×90)	140 (12×16)
6	325546 (496×661)	19886 (124×165)	336 (18×23)
7	605926 (676×901)	54540 (204×272)	660 (24×32)
8	638716 (694×925)	112230 (292×389)	1200 (32×42)

instead of 8), is more restrictive than in the matrix organisation case: as can be seen in figure 3.8 the Hamming distance that can be reached for a pattern of a suitable size is lower. In figure 3.8 a 4:3 aspect ratio was used, and the columns are closer together than the rows as explained before, leading to a factor of $\frac{8}{3\sqrt{3}}$ ($\approx 54\%$) more columns than rows.

3.2.7 Results: generated patterns

Table 3.1 contains the results of the proposed algorithm for perfect map generation. All results are obtained within minutes (exceptionally hours) on standard PCs. The table shows the number of potentially [1] reconstructed points and between brackets the size of the 2D array. A $*$ indicates that the search space was calculated exhaustively and no array with a bigger size can be found.

To test the influence of the rotational invariance constraint, we ran the same algorithm without those constraints. Logically, this produces larger patterns, see table 3.2. For a large a and $h = 1$, the algorithm constantly keeps finding larger patterns, so there the size of the pattern is dependent on the number of calculation hours we allow, and in this case comparing table 3.1 and 3.2 is not useful. These patterns are anyhow by far large enough.

We compare the results to those published by Morano et al. [1998]. We use our algorithm without the rotational constraints, and specify that the pattern

[1] potentially because the camera does not always observe all spots

Table 3.3: Code and pattern sizes for hexagonal patterns with rotational independence constraint

$a\backslash h$	1	3	5
2	24 (6×8)	imposs.	imposs.
3	150 (12×17)	12 (5×6)	imposs.
4	1683 (35×53)	35 (7×9)	2 ($3 \times 4^*$)
5	4620 (57×86)	54 (8×11)	6 (4×5)
6	9360 (80×122)	54 (8×11)	6 (4×5)
7	10541 (85×129)	54 (10×14)	12 (5×6)
8	11658 (89×136)	96 (10×14)	12 (5×6)

Table 3.4: Code and pattern sizes for hexagonal patterns without rotational independence constraint

$a\backslash h$	1	3	5
2	77 (9×13)	imposs.	imposs.
3	748 (24×36)	24 (6×8)	imposs.
4	7739 (73×111)	77 (9×13)	2 ($3 \times 4^*$)
5	16274 (105×160)	176 (13×18)	6 (4×5)
6	20648 (118×180)	551 (21×31)	6 (4×5)
7	23684 (126×193)	805 (25×37)	12 (5×6)
8	31824 (146×223)	1276 (31×46)	12 (5×6)

needs to be square. Our algorithm generates bigger arrays with smaller alphabets than the one by Morano et al. Morano et al. indicate which combinations of a and h are able to generate a 45×45 array. For example, with $h = 3$ Morano et al. need an alphabet of size $a = 8$ or larger to generate a 45×45 perfect map. Our approach reaches 38×38 for $a = 4$, and 78×78 for $a = 5$ already. For $h = 2$, Morano et al. need $a = 5$ for such perfect map, our algorithm reaches 40×40 for $a = 3$, and 117×117 for $a = 4$. Another advantage of our approach is that results are reproducible, and not dependent on chance, as in the approach using random numbers by Morano et al..

Table 3.3 displays the results for the hexagonal variant. As explained before, this configuration is more restrictive, so logically, the patterns are not as large as in the non-hexagonal variant.

If we remove the rotational constraints, patterns remain relatively low in size. This is logical, since each submatrix has less neighbours (only 6) than in the non-hexagonal (matrix structure) variant, where there are 8 neighbours. Thus there are 2 DOF less: less workspace to find a suitable pattern. The results are in table 3.4.

3.2.8 Conclusion

We present a **reproducible**, deterministic algorithm to generate 2D patterns for single shot structured light 3D reconstruction. The patterns are **independent** of the relative **orientation** between camera and projector and use

error correction. They are also large enough for a 3D reconstruction in robotics to get a general idea of the scene (in sections 3.4, 8.3 and 8.4 it will become clear how to be more accurate if necessary). The pattern constraints are more restrictive than the ones presented by Morano et al. [1998], but still the resulting array sizes are superior to it for a fixed alphabet size and Hamming distance.
Instead of organising the elements in a matrix, one could also use a hexagonal structure. An advantage of a hexagonal structure is that it is more dense. However, due to the limited number of neighbours (6 instead of 8) the search algorithm does not find patterns as large as in the matrix structure case for a fixed number of available letters in the alphabet, and a fixed minimal Hamming distance. Or in other words, applied to robotics, given a fixed pattern size needed for a certain application, the number of letters in the alphabet needed is larger, and/or the Hamming distance is smaller for the hexagonal case than for the matrix form. Therefore, the rest of this thesis will continue working with the perfect maps like in figure 3.7.

3.3 Pattern implementation

3.3.1 Introduction

Often, patterns use different colours as projected features like the pattern of figure 3.6 c, but using colours is just one way of implementing the patterns of section 3.2. Other types of features are possible, and all have their advantages and disadvantages.

Redundancy

The aim of this subsection is to show the difference between the minimal information content of the pattern, and the information content of the actual projection. This redundancy is necessary because of the deformations of the signal because of scene colours and shapes. Other than that, this subsection is not strictly necessary for the understanding of the rest of section 3.3.1 and the sections thereafter.

In terms of information theory, the feature implementations are representations of the alphabets needed for the realisation of the patterns of section 3.2. We can for example choose to represent the alphabet as different colours.
Since the transmission channel is distortive, the only way to get data across safely, is to add redundant information. The entropy (amount of information) of one element at a certain location in the pattern is:

$$H_{i,j} = -\sum_{k=0}^{a-1} P(\mathbf{M}_{i,j} = k) \log_2(P(\mathbf{M}_{i,j} = k))$$

with a the number of letters in the alphabet, $\mathbf{M}_{i,j}$ the code of matrix coordinate (i, j) in the perfect map, and $P(\mathbf{M}_{i,j} = k)$ the probability that that code is the letter k.

For example, we calculate the entropy for the pattern on the top right of figure 3.7 ($a = 5$). The total number of projected features is $rc = 36 \cdot 48 = 1728$. Let n_i be the number of occurrences of letter i in the pattern. Assuming the value of each of the elements is independent of the value of any other, the entropy (in bits) of one element is:

$$H_{i,j} = -\sum_{k=0}^{4} \frac{n_k}{rc} \log_2(\frac{n_k}{rc}) = 2.31b$$

with $n_0 = 410$, $n_1 = 379$, $n_2 = 334$, $n_3 = 320$, $n_4 = 285$ (simply counting the number of features in the pattern at hand). The amount of information for the entire pattern is then:

$$H = \sum_{i=0}^{r-1} \sum_{j=0}^{c-1} H_{i,j} = r\ c(231b) = 3992b$$

Figure 3.1 presented an overview of the structured light setup as a communication channel. H is the amount of information in the element "pattern" of this figure. We multiplex this information stream with the information stream of the scene. We now determine the amount of information after the multiplexing, as seen by the camera if each element of the pattern would be represented by a single ray of light, using a different pixel values. Let $\mathcal{W}_c$ be the width in pixels of the camera image, and $\mathcal{H}_c$ the height. The probability that none of the projected rays can be observed at a certain camera pixel $Img_{u,v}$ is $P(Img_{u,v} = a) = \dfrac{\mathcal{W}_c\mathcal{H}_c - rc}{\mathcal{W}_c\mathcal{H}_c}$. The probability that a letter k ($k = 0..a-1$) is observed at a certain camera pixel is

$$P(Img_{u,v} = k) = \frac{rc}{\mathcal{W}_c\mathcal{H}_c} \frac{n_k}{\sum_{l=0}^{a-1} n_l} = \frac{n_k}{\mathcal{W}_c\mathcal{H}_c}$$

The entropy of camera pixel $(u,\ v)$ is:

$$H_{u,v} = \sum_{k=0}^{a} -P(Img_{u,v} = k) \log_2(P(Img_{u,v} = k)) = 0.06b$$

Then the entropy of the patterns multiplexed with the scene depths is $H' = \mathcal{W}_c\mathcal{H}_c(0,06b) = 19393b$ Thus the theoretical limit for the compression of this reflected pattern is 2425 bytes.

Comparing this value with the information content of the received data stream of the camera illustrates the large amount of redundancy that is added, in order to deal with external disturbances and model imperfections. For a grey scale VGA camera this is $640 \cdot 480 \approx 3.10^5$ bytes.

Implementation requirements

Section 3.2 chose to generate patterns in which neighbouring elements can be the same letters of the alphabet. This would otherwise be an extra constraint that would limit the pattern sizes. Therefore in this pattern implementation section we discuss the corresponding possibilities for pattern elements between which there is unilluminated space. This is the 2D extension of the 1D pattern concept of **multi-slit** patterns. The other possibility is to use a continuously illuminated pattern, without dark parts in between, see [Chen and Li, 2003, Chen et al., 2007]. That is the 2D equivalent of the 1D concept of stripe patterns. Chen et al. [2007] calls this continuously illuminated patterns grid patterns. Unfortunately, the same name is given by Salvi et al. [1998] and Fofi et al. [2003] to their patterns, see the bottom right drawing of figure 3.6.
Concluding, to avoid the extra code constraint that is associated with continuously lit patterns, this section discusses pattern implementations with as a global projection strategy **spatial encoding, with stand-alone elements separated by non-illuminated areas**.

This section discusses projector implementation possibilities keeping robot arm applications in mind (e.g. low resolution patterns). The reflection of the chosen pattern will then be segmented in chapter 5.2: this thesis chooses an implementation such that the data is correctly associated under as many circumstances as possible. Robustness is our main concern here, then accuracy and only after that, resolution. For robustness' sake, it is wise to:

- *choose the representations of the letters of the alphabet to be as far apart as possible.* Then the probability to distinguish them from one another is maximised. For example, if we choose colours as a representation and need three letters in the alphabet, a good choice would be red, green and blue, as their wavelengths are well spread in the visible spectrum.
- *avoid threshold-dependent low level vision algorithms.* Many low level vision algorithms depend on thresholds (e.g. edge detection, segmentation, ...). One prerequisite for a robust segmentation is that there are ways to circumvent these thresholds. We'll choose the projection features such that fixed thresholds can be converted into adaptive ones, or threshold-free algorithms can be used.
- *use compact shapes.* For all encoding techniques but the shape based one, we use a filled circular shape for each element. This is a logical choice: it is the most compact shape, segmentation is more reliable as more illuminated pixels are present within a predefined (small) distance from the point to be reconstructed. Hence, of all shapes a circle performs best.

The sections that follow discuss the implementation of a single projective element in the pattern: the temporal encoding of section 3.3.4 and the spatial one of section 3.3.5. This is not to be confused with temporal and spatial global projection strategies as discussed in 3.2.1. Often, combinations of these implementations are also possible, for example colour and shape combined.

3.3.2 Spectral encoding

Most of the recent work on single shot structured light uses an alphabet of different colours, see for example [Morano et al., 1998], [Adán et al., 2004], [Pagès et al., 2005] and [Chen et al., 2007]. Figure 3.9 illustrates this for one of the smaller patterns (to keep the figure small and clear) resulting from section 3.2. In order to reduce the influence of illumination changes, segmentation should be done in a colour space where the hue is separated from the illuminance. HSV and Lab are examples of such spaces. RGB space on the contrary does not separate hue and illuminance, and should be avoided, as section 5.2 explains.

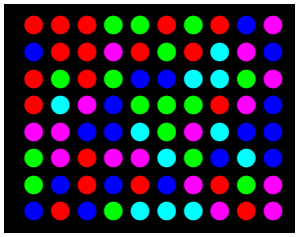


Figure 3.9: Spectral implementation of pattern with $a = h = 5$, $w = 3$

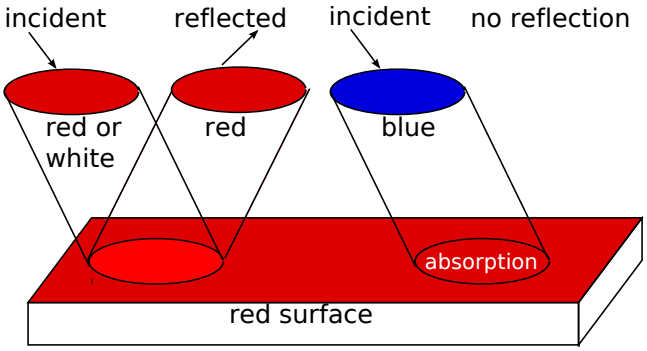


Figure 3.10: Selective reflection

For (near) white objects this works fine, since then the whole visible spectrum is reflected and thus any projected colour can be detected. But applying this technique on a **coloured surface does not work** (without extra precautions), since being coloured means only reflecting the part of the visible spectrum corresponding to that colour, and absorbing the other parts. In other words, colour is produced by the absorption of selected wavelengths of light by an object and the reflection of the other wavelengths. Objects absorb all colours except the colour of their appearance. This is illustrated in figure 3.10: only the component of the incident light that has the same colour as the surface is reflected. For example a red spot is reflected on a red surface, but a blue spot is absorbed. In this case, we might as well work with white light instead of red light, as all other components of white are absorbed anyway.

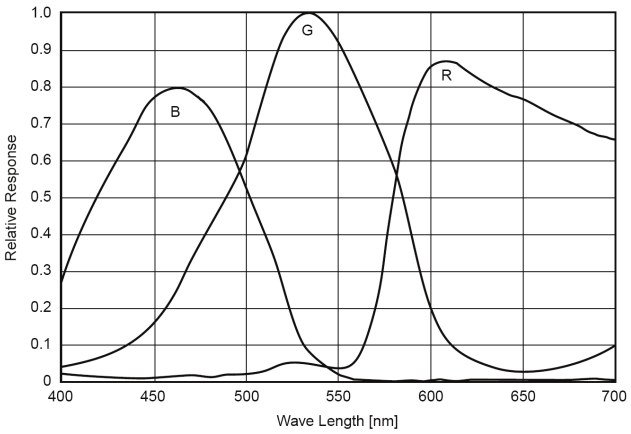


FIGURE 3.11: Spectral response of the AVT Guppy F-033

However, it is possible to work with coloured patterns on coloured surfaces, by performing a **colour calibration** and adapting the pattern accordingly. It is then necessary to perform an intensity calibration (as in section 4.2) for the different frequencies in the spectrum (usually for 3 channels: red, green and blue) and account for the **chromatic crosstalk**. The latter is the phenomenon that light that was e.g. emitted as red, will not only excite the red channel of the camera, but also the blue and green channel: the spectral responses of the different channels overlap. This is illustrated in figure 3.11 for the camera with which most of the experiments of chapter 8 have been done. A synonym for chromatic crosstalk is spectral crosstalk.

Caspi et al. [1998] is the first to use a light model that takes into account the spectral response of camera and projector, the reflectance at each pixel, and the chromatic crosstalk. After acquiring an image in ambient light (black projection image), and one with full white illumination, the colour properties of the scene are analysed. Caspi et al. [1998] locates, for each colour channel (R, G, B), the pixel where the difference in reflection between the two images in that channel is the smallest. This is the weakest link, the point that puts a constraint on the number of intensities that can be used in that colour channel. Given a user defines the minimal difference in intensity in the colour channels to be able to keep the colour intensities apart, the number of letters available within each of the colour channels can be calculated. The pattern is thus adapted to the scene. For coloured patterns in colourful scenes, adaptation of the pattern to the scene is the only way for spectral encoding to function properly. For a more detailed survey, see [Salvi et al., 2004].

The model of Caspi et al. [1998] only contains the non-linear relation of the projector between the requested and the resulting illuminance. A similar relation for the camera is not present (a projector can be seen as an inverse camera). She does consider the spectral response of the camera but does not integrate the

spectral response of the projector. Both of these incompletenesses are corrected by Grossberg et al. [2004]. Koninckx et al. [2005] suggests to write the reflection characteristic and the chromatic crosstalk separately: although mathematically equivalent (matrix multiplication), this is indeed a physical difference. Grossberg et al. [2004] uses a linear crosstalk function, whereas in [Koninckx et al., 2005] it is non-linear. Since 3.11 shows that the correlation between wavelength and intensity response is non-linear, one can indeed gain accuracy by also making the model non-linear.

Concluding, when the pattern is implemented using colours, we need to either restrict the scene to white or near white objects (that reflect the entire visible spectrum), or adapt the pattern to the scene. If we want to be able to reconstruct a coloured scene, the frequency of the light at each spot must match the capability of the material at that spot to reflect that frequency. Red light for example is absorbed by a green surface.
Thus, we **need several camera frames** to make one reconstruction in order to make a model of the scene before a suitable pattern can be projected onto it. Since we want to make a single frame reconstruction of a possibly coloured scene, the main pattern put foward by this thesis does not use colour information. In that case one only needs to perform an intensity calibration, not a colour calibration.

3.3.3 Illuminance encoding

This thesis tries to deal with as diverse scenes as possible. Hence, it assumes that the scene could be colourful. If we want to use coloured patterns, a system such as presented by Caspi et al. [1998] is necessary. But then also several frames during which the scene must remain static, at least three: with ambient light, with full illumination and with the adapted coloured pattern. Therefore, if we want the scene to be able to move, we cannot use colour encoding.

Illuminance encoding only varies the intensity of the grey scale pattern. If one wants to avoid the constraint of a static scene during several patterns, one needs to project all visible wavelengths. The maximal illumination of light bulb in the projector used for the experiments is 1500 lumen [2], which can be attenuated at any pixel using the LCD.

Optical crosstalk and blooming

Even using only intensities and not colours, **optical crosstalk** is a problem. Optical crosstalk is the integration of light in pixels that are neighbours of pixels the light is meant for. Reflection and refraction within the photosensor structure can give rise to this stray light. Hence bright spots appear bigger than they are in the camera image. It is slightly dependent on wavelength, but more so on pixel pitch. A synonym for optical crosstalk is spatial crosstalk. The cameras used for the experiments of chapter 8 all have a CCD imaging sensor: CCD has a lower optical crosstalk than CMOS sensors [Marques and Magnan, 2002]. However, CCD has other problems.

CCD bloom is a property of CCD image sensors that causes charge from the potential well of one pixel to overflow into neighbouring pixels. It is an overflow of charge from an oversaturated pixel to an adjacent pixel. It is thus oversaturation that will cause the imperfections to become visible. As a result, the bright regions appear larger than they are. Hence it is wise to adapt the shutter speed of the camera such that no part of it is oversaturated and blooming is reduced. CMOS does not have this problem. Blooming is due to the fact that lenses never focus perfectly. Even a perfect lens will convolve the image with an Airy disc (the diffraction pattern produced by passing a point light source through a circular aperture).
Manufacturers of image sensors try to compensate for these effects in their sensor design. These effects are hard to quantify, one needs for example Monte Carlo simulation to quantify it.

In order to make the segmentation more robust, the intensities need to be as diverse as possible. Hence, we will use the full illumination of the projector next to an area with only ambient light. This makes the problem more pronounced: the projected elements will appear larger than they are. The usual solution to find the correct edges in the image is to first project the original pattern, and then its inverse, the average of the two edges is then close to the real edge. But this would be compromising the single shot constraint, and thus the movement

[2] a Nec VT57

of the scene. Not to mention that such a flickering pattern would be annoying for the user. So the pattern we select should not depend on the precise location of these edges. Indeed, if only the centre of the projected element is important and not its size, there is no problem.

When Salvi et al. [2004] discusses techniques based on binary patterns, he mentions that two techniques exist to detect a stripe edge with subpixel accuracy. One is to find the zero-crossing of the second derivative of the intensity profile. The other one is to project the inverse pattern and then find the intersection of the intensity profile of the normal and inverse patterns. Salvi et al. concludes that the second one is more accurate, but does not mention a reason. If one has the luxury of being able to project the inverse pattern – then the scene has to remain static during 2 frames – optical crosstalk and blooming have their effect in both directions: the average is close to the real image edge. In figure 3.12, the left and right dashed lines are the zero-crossings of the intensity profile: both are biased. The crossing of both profiles however are a better approximation, as the average crosstalk error is 0.

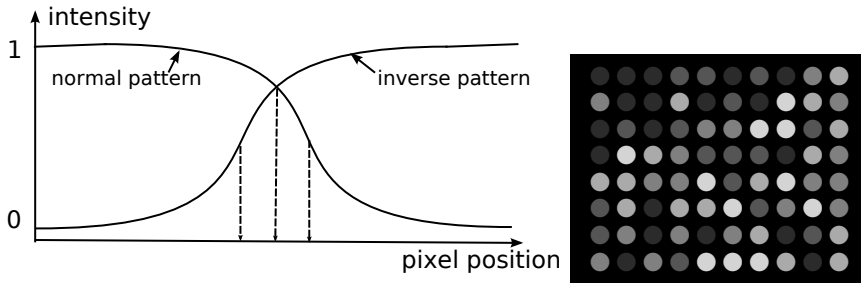


Figure 3.12: Left: effect of optical crosstalk/blooming on intensity profiles, right: illuminance implementation of pattern with $a = 5$, $h = 5$, $w = 3$

3.3.4 Temporal encoding

As an alternative to colour coding, Morano et al. [1998] suggest to vary each element of the pattern in time.

3.3.4.1 Frequency

For example one can have the intensity of the blobs pulsate at different frequencies and/or phases, and analyse the images using FFT. Instead of the intensity one could also vary any other cue, the hue for example.
In figure 3.13 only the frequency is varied according to:

$$I(i,j,t) = I_{min} + \frac{1 - Imin}{2}(\sin(\frac{2\pi s f_r(c_{i,j}+1)t}{2a}) + 1)$$

for $i = 0..r-1$, $j = 0..c-1$. $0 \leq c_{i,j} \leq a-1$. I_{min} is the minimal projector brightness that can be segmented correctly. $\forall i,j,t : 0 \leq I_{min}, I(i,j,t) \leq 1$ with 1 the maximum projector pixel value. f_r is the frame rate of the camera and s is the safety factor to stay removed from the Nyquist frequency $\frac{f_r}{2}$. As stated by the Nyquist theorem, any frequency below half of the camera frame rate can be used. For example, for an alphabet of 5 letters, and a 15fps camera: blobs pulsating at 1,2,3,4 and 5 Hz are suitable, keeping a safety margin from the Nyquist frequency where aliasing begins.
Figure 3.13 shows this pattern implementation for lower frequencies to demonstrate it more clearly: for $f_r = 4Hz$ the four displayed patterns are projected in one second (with $s = 0.8$). Since $\forall i,j : t = i\pi \Rightarrow I(i,j,t) = 0$, the figure does not display the states at $t = 0s$ or $t = 0.5s$ as all features are then equal to I_{min}. Instead, from left to right the states at $t = \frac{1}{8}s$, $\frac{3}{8}s$, $\frac{5}{8}s$ and $\frac{7}{8}s$ are shown.

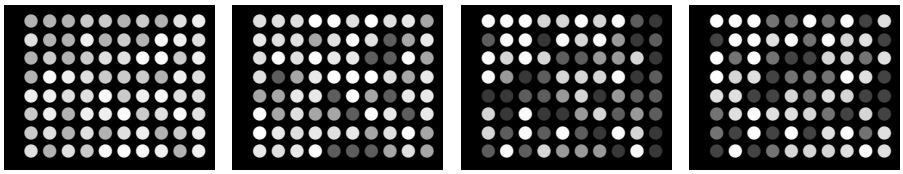


FIGURE 3.13: Temporal implementation of pattern with $a = 5$, $h = 5$, $w = 3$: different frequencies

Segmentation is by comparing the lengths of the DFT vectors in the frequency domain. The longest vector defines the dominant frequency.

3.3.4.2 Phase

Analogously, the phase can be used as a cue. In figure 3.14 only the phase is changed, according to:

$$I(i,j,t) = I_{min} + \frac{1 - Imin}{2}(\sin(2\pi(\frac{c_{i,j}}{a} + t)) + 1)$$

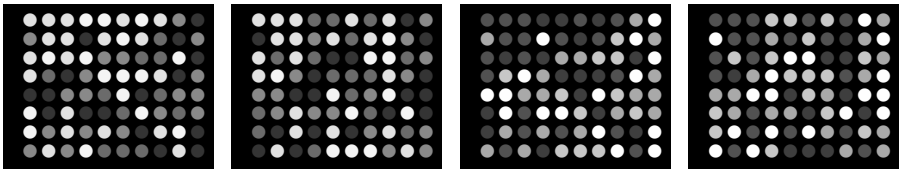


Figure 3.14: Temporal implementation of pattern with $a = 5$, $h = 5$, $w = 3$: different phases

Figure 3.14 shows phase shifted patterns. The phase shifting can be segmented by determining the angle between the DFT vectors and the real axis (tangent of rate of the imaginary to the real part of the DFT coefficients). Clearly, one can also choose for a combination of frequency variation and phase shifting. The number of discretisation steps that is needed in each of them is then smaller, making the segmentation more robust. Let the number of discretisation steps of the frequency be n_f and the one for the phase n_p. As these are orthogonal visual cues, n_f and n_p can be chosen as small as $n_f n_p \geq a$ allows. Instead of this (almost) continuous change, more discrete changes are also possible: a sequence of discontinuously changing intensities, as is used in binary Gray code patterns [Inokuchi et al., 1984].

3.3.4.3 Limits to scene movement

Using temporal encoding however, the scene has to remain static from the beginning to the end of each temporal sequence. Or, if one tracks the changing dots, the system can be improved such that each dot can be identified at any point in time, since its history is known due to the tracking. Hall-Holt and Rusinkiewicz [2001] for example presents a system where *slow* movements of the scene are allowed: fast movements would disable tracking. Since we would like to avoid constraints on the speed of objects in the scene, this implementation is avoided.

3.3.4.4 Combination with other cues to enhance robustness

These temporal techniques change the pattern several times to complete the code of each of its elements. However, if the pattern at any point in time contains the complete codes, it is interesting to add a temporal encoding on top of that, to add redundancy. Even if one chooses the pattern such that it is unlikely that it will be confused with a natural feature, that chance can never be completely excluded. However, changing the pattern over time can reduce this probability further. For example, if the pattern uses a codes, shift the implementation the codes each second, as a cyclic permutation: the implementation of code 0 changes to code 1, code 1 changes to code 2, ... and code $a - 1$ changes to code 0. Then one can perform the extra check whether the projected pattern change reflects in the corresponding change in the camera image. One can apply this technique to the patterns of sections 3.3.2, 3.3.3 and 3.3.5. Section 3.3.6 discusses the pattern we choose for the experiments: there also adding this type of temporal encoding makes the codec slightly more complex but increases robustness.

3.3.5 Spatial encoding

Choosing different shapes for the elements of the pattern is another possibility.

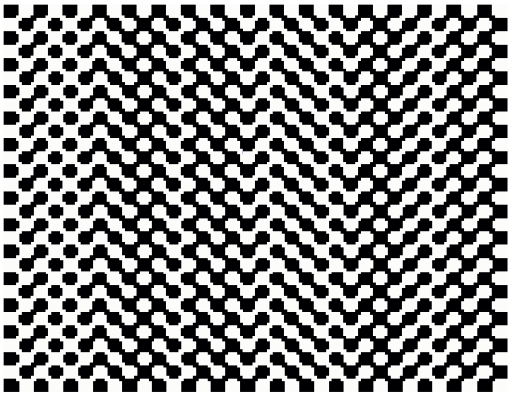


FIGURE 3.15: 1D binary pattern proposed by Vuylsteke and Oosterlinck

An example of this is the pattern by Vuylsteke and Oosterlinck [1990]: it has binary codewords that are encoded by black or white squares, see figure 3.15. This pattern has no error correction (only error detection: it has Hamming distance h=2), features no rotation invariance, and is not 2D: it only encodes columns. The first two properties are only desired for the robot application studied here, but the last one is required. Therefore we do not continue using this pattern. Salvi et al. [2004] summarizes this technique.

Shape based

A simple way to keep shapes apart is using their **perimeter efficiency** k [Howard, 2003]. It is a dimensionless index that indicates how efficient the perimeter p is spanned around the area A of the shape: $k = \frac{2\sqrt{\pi A}}{p}$. The normalisation factor $2\sqrt{\pi}$ makes $k = 1$ in case of a circle. Another name for the same concept is the *isoperimetric quotient* $Q = k^2$. The isoperimetric inequality states that for any shape $Q \leq 1$: of all shapes, a circle has the largest perimeter efficiency.

For regular polygons with n sides $k = \sqrt{\frac{\pi}{n \tan \frac{\pi}{n}}}$. For example, for an equilateral triangle $k = 0.78$, for a square $k = 0.89$. In principle, regular polygons with more sides can also be used, but their perimeter efficiency is too close to 1: they might be taken for a circle while decoding. Especially since the scene geometry deforms the projected shapes. At a surface discontinuity, part of e.g. a triangle may be cut in the camera image, giving it a perimeter efficiency closer to a square than a triangle. Therefore, this method is not so robust.

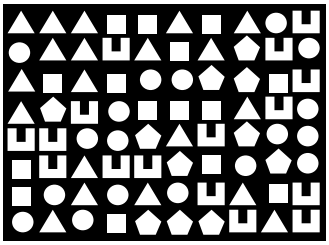


Figure 3.16: Shape based implementation for a pattern with $h = a = 5$, $w = 3$

A more refined way to characterise shapes, is to use **Fourier descriptors**, see [Zhang and Lu, 2002]. A Fourier descriptor is obtained by applying a Fourier transform on a shape signature. The set of normalised Fourier transformed coefficients is called the Fourier descriptor of the shape. The shape signature is a function $r(s)$ derived from the shape boundary pixel coordinates, parametrised by s. A possible function is to use the u and v coordinates as real and imaginary part: $r_1(s) = (u(s) - u_c) + i(v(s) - v_c)$ where (u_c, v_c) is the centroid of the shape. Another possibility is to use distances: $r_2(s) = \sqrt{(u(s) - u_c)^2 + (v(s) - v_c)^2}$. Zhang and Lu [2002] conclude that the centroid distance function $r_1(s)$ is the optimal shape signature in terms of robustness and computational complexity. These descriptors are translation, rotation and scale invariant. The result is a series of numbers that characterise the shape, and can be compared to other series of numbers. As more of these numbers are available than in case of the perimeter efficiency (which is only one number), confusing shapes is less likely. This thesis implemented and tested this for triangles, squares and circles, with satisfying recognition results on continuous shapes. Problem remains that if discontinuities of the scene cut part of the shape, the features at the discontinuity become unrecognisable. The size of the shapes should be as large as possible to recognise them clearly, and as small as possible to avoid discontinuity problems and to precisely locate its centre: a balance between the two is needed. Hence, we will not make the shapes larger than is needed for recognition in the camera image: section 3.4 will explain how the size of the shapes in the projector image is adapted to its size in the camera image.
As noted before, the most compact shape is a circle, so any other shape somewhat compromises this balance: the probability to decode the feature erroneously increases for a constant number of feature pixels.

Spatial frequencies

Pattern layout This section presents a previously unpublished pattern that encodes the letters in a circular blob. The outer edge is white, to make blob detection easier, but the interior is filled with a tangential intensity variation according to one or more sine waves. The intensity variation is tangential and not radial to ensure that every part of the sine wave has an equal amount of pixels in the camera image, increasing segmentation robustness. Thus, we use an

analogue technique here (sinusoidal intensity variations). Figure 3.17 presents this pattern.

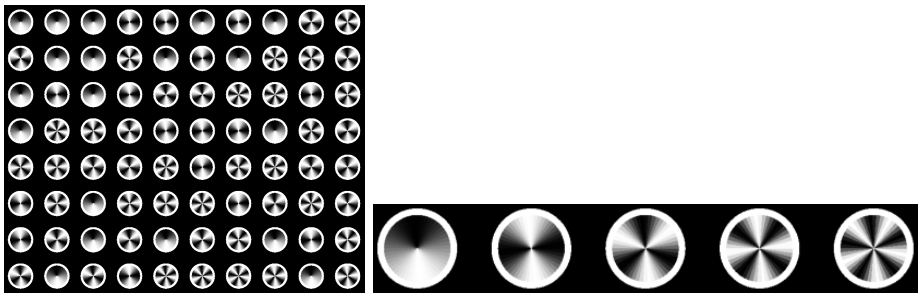


FIGURE 3.17: Spatially oscillating intensity pattern. Left: projector image for a pattern with $h = 5$, $a = 5$, $w = 3$, right: individual codes

Segmentation An FFT analysis determines the dominant frequency for every projection element. These sine waves are inherently redundant. This has the advantage that the probability to confuse this artificial feature with a natural one is drastically reduced. Whereas if we use blobs in one colour or intensity, incident sunlight in a more of less circular shape for example might easily be taken for one of the projected blobs.
In the decoding chapter, chapter 5.2, we discuss the segmentation of a different kind of pattern, the pattern explained in section 3.3.6. The decoding of this spatial frequency pattern will be explained here in this section, to avoid confusion between both patterns in the decoding chapter. Currently, this pattern implementation is not incorporated in the structured light software, although it would also be a good choice. A step by step decoding procedure:

- Optionally downsample the image to accelerate processing.
  Image pyramids drastically improve performance. During the stages of the segmentation, use that image from the pyramid whose resolution is best adapted to the features to be detected at that point. This limits the segmentation of redundant pixels. For example, to find the outer contours of the blobs, it's overkill to use the full resolution of the image. The processing time needed to construct the pyramid is marginal compared to the time gained during decoding using the different levels of the pyramid.
- Conversion from grey scale to a binary image, using a threshold that is calculated online based on the histogram: this is a preprocessing step for the contour detection algorithm.
- The Suzuki-Abe algorithm [Suzuki and Abe, 1985] retrieves (closed) contours from the binary image by raster scanning (following the scan lines) the image to look for border points. Once a point that belongs to a new border is found, it applies a border following procedure.
- Assume that the scene surface lit by every projection element is locally planar. Then circles in the projection image transform to ellipses in the

camera image. Fit the recovered contours to ellipses, if fitting quality is insufficient, reject the blob since it's probably not a projected feature then. This extra control step increases robustness.

- To calculate the coefficients of the FFT, the pixel values need to be sorted by their angle in the ellipse. One could simply sort all pixels according to their angle. Or, in order to improve the efficiency and give somewhat in to the robustness, take pixel samples from the blob from a limited number of angles. According to the Nyquist-Shannon sampling theorem the sampling frequency (the number of angles) should be more than double the maximum frequency of the sinuses in the blobs. In the case represented by figure 3.17 the maximum is 5 periods: sampling should be faster than 10 samples/2π to avoid aliasing, preferably with a safety margin. The pixels needed are those from the original grey scale image. Do not include the pixels near the outer contour (those are white, not sinusoidal), nor the ones near the centre (insufficient resolution there).
- Use the median (preferably not the average) of every discretisation box to produce a 1D signal for every blob.
- Perform a Discrete Fourier transform for every blob. The length of the vector in the complex plane of the frequency domain is a measure for the presence of that frequency in the spatial domain.
- Label every blob with the frequency corresponding to the longest of those vectors.

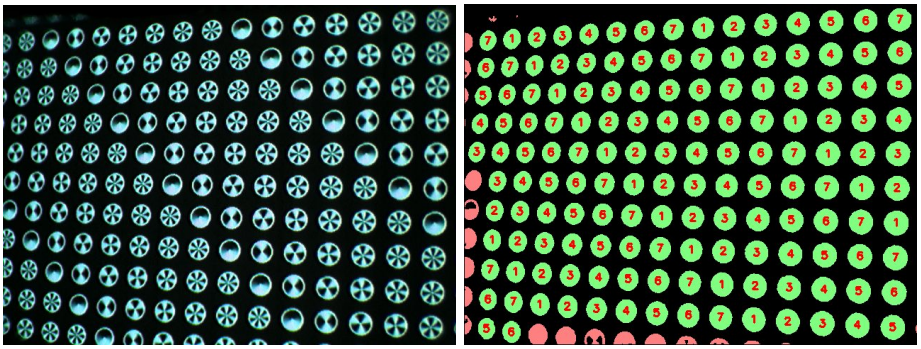


Figure 3.18: Spatially oscillating intensity pattern: camera image and decoding

The results of this segmentation experiment are satisfactory, as can be seen for a test pattern in figure 3.18. We assume the **reflectance characteristic** at each of the elements of the pattern to be **locally constant**. The smaller the feature, the more valid this assumption is, but the harder it is to segment it correctly. The reflection models of section 4.2 are relevant here, as one of the advantages of this technique is that the absolute camera pixel values are unimportant. Only the relative difference to their neighbouring pixels is. Mind that the shutter speed of the camera avoids oversaturation.

3.3.6 Choosing an implementation

Concentric circles

Shape based and spectral encodings are not very robust as explained above, and temporal encoding restricts the scene unnecessarily to slowly moving objects. Therefore two options remain: to use grey scale intensities, or to use the spatial frequencies (also in grey scale) to implement the pattern.

The amount of reflected light is determined both by the intensity of the projector, and the reflection at each point in the image. Before the correspondences are known, we cannot estimate both at the same time. Therefore we **include a known intensity** in every element of the pattern: an intensity that almost saturates the camera (near white in the projector). Hence, each element of the pattern needs to contain at least two intensities: (near) white and a grey scale value. The more **compact** one can implement these, the more often the local reflectance continuity assumption is valid, and the less problems with depth discontinuities. Two filled **concentric circles** is the most compact representation.

In the **spatial frequency** case, the blobs had a white outer rim for easy background segmentation. This thin white belt appears larger in the camera image, as it induces optical crosstalk. This is fine for a border detection, but makes it hard to measure the brightness of the pixels this rim induces in the camera image. One would need to expand this rim to an amount of pixels that is sufficient to robustly identify the corresponding pixel value in the camera image. This reduces the amount of pixels available for the spatial frequencies to levels where the frequency segmentation becomes difficult. Or, if one decides to keep this part of the blob large enough, leads to a **larger blob** that will more often violate the assumption of local reflectance continuity, and more importantly, suffer more often from depth discontinuities. Another advantage of having two even intensity parts in comparison to the spatial frequency pattern, is that it is **less computationally expensive**, as one does not need to calculate a FFT.

Thus, assume a blob of two even intensity parts. These **intensities** are both linearly attenuated when the dynamic range of the camera demands it, when the pixels saturate. This linear attenuation is an approximation for two reasons.

- The Phong reflection model, explained in more detail in section 4.2, states that the amount of reflected light is proportional to the amount of incident light, in case one neglects the ambient light.
- For an arbitrary non-linear camera or projector intensity response curve, a linear scaling of both intensities does not result in a linear scaling of the responses. But when one can approximate the function with a quadratic function, the proportions do remain the same.

More formally, let g_p and g_c be the response curves of the camera and projector. Then according to the Phong model $g_c(I_c) = I_{ambient} + Cg_p(I_p)$ with C some factor. Approximate $g_c(I_c)$ as $c_c I_c^2$ and $g_p(I_p)$ as $c_p I_p^2$. Neglecting $I_{ambient}$ results in: $I_c \sim I_p$, hence the linear attenuation.

Thus, the constraints to impose to these concentric circles are:

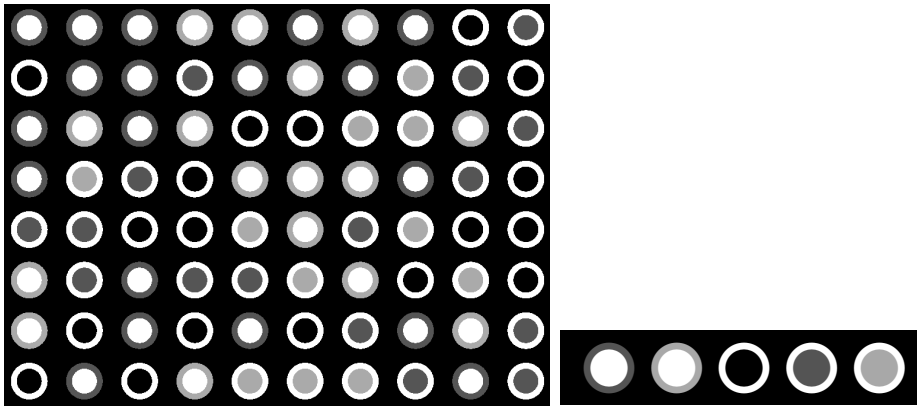


Figure 3.19: Left: pattern implementation with concentric circles for $a = h = 5$, $w = 3$; right: the representation of the letters $0, 1, 2, 3, 4$

- The outer circle cannot be black, since there would be no difference with the **background** any more then.
- Either inner or outer circle should be white, for the reflectance adaptation.
- For maximum robustness the **number of pixels** of inner and outer parts should be **equal**. The radius of the inner circle is thus a factor $\frac{1}{\sqrt{2}}$ of the radius of the outer circle.

Intensity discretisation

Suppose we can use a' different intensity levels, the first one black, and the last one white. One of both parts needs to be white. If it is the outside part, $a' - 1$ possibilities remain for the inside (the intensity cannot be the same). If it is the inside part, $a' - 2$ possibilities remain for the outside. The outside ring cannot have the same intensity as the inside circle, and cannot be black. In total: $2a' - 3$ possibilities, or letters of an alphabet. The smaller a', the more robust the segmentation. Try $a' = 3$, then the pattern can represent $a = 3$ letters. If one wants to be able to correct one error, according to table 3.1, the size of the pattern is only 9×12. Therefore choose $a' = 4$, resulting in $a = 5$ letters, as shown in figure 3.19. Chapter 5.2 decodes (a larger variant of) this pattern.

3.3.7 Conclusion

The proposed pattern makes the data association between features in projector and camera image as trustworthy as possible. It is robust in multiple ways:

- *False positives are avoided*: there should be as little probability as possible to confuse projected features with natural ones.
  This problem is mainly solved by the large intensity of the projected features. Ambient light resulting from normal room lighting is filtered out: the parts of the image where only this light is reflected are dark in the camera image. The camera aperture suitable for the projector features reduces all parts of the image that do not receive projector light to almost black. However, sunlight is bright enough to cause complications, even far brighter than the projector light. Therefore, if the incident sunlight happens to be ellipse-shaped, it could be interpreted as a projected feature. However, the probability that within this ellipse, another concentric ellipse with a different intensity is present at a radius of about 70% of the original one, is very low. Hence, we perform these checks online, as explained in chapter 5.2. This is comparable to redundancy added when storing data: error correcting codes like the ones of Reed-Solomon or BCH.

- *Different reflectances are accounted for.*
  Different colours and material reflection characteristics require a different behaviour of the projector in each of the projected features. Reflectivity is impossible to estimate unless all other parameters of the illumination chain are known. The source brightness and surface reflectivity are multiplexed and cannot be calculated unless one of them is known. Hence, we make sure that part of the projected element always contains a known projector intensity value (white in this case, possibly attenuated to avoid for camera saturation). This part is recognisable, since it is brighter than the other part: the system will only work with **relative** brightness differences, as they are more robust than absolute ones.

- *(Limited) discontinuity in scene geometry is allowed*
  Any shape of pattern is allowed, as long as most of it is locally continuous. The projected features have a minimal size to be detected correctly, and the intersection between the spatial light cone corresponding to every single feature and the surface should not be discontinuous, or have a strong change in surface orientation. Since the pattern only uses local information (a 3 by 3 window of features), a depth discontinuity (or strong surface orientation change) only influences the reconstruction at this discontinuity (or orientation change). This is unlike structured light with fixed gratings without individually recognisable elements: for more details on this difference, see [Morano et al., 1998]
  The capability of correcting one error in each codeword helps to compensate for faulty detections at depth discontinuities.
  Moreover, the system is not limited to polyhedral shapes in the scene, as

would be the case in the work of [Salvi et al., 1998]. He uses a grid like the bottom right drawing of figure 3.6. Straight segments should remain straight in the camera image, because they need to be detected by a Hough transform. Therefore his system cannot deal with non-polyhedral scenes.

- *Scene movement*: the pattern is one-shot, so scene movement is not a problem. Its speed is only limited by the shutter speed of the camera. This allows for reasonably fast movements, as we will illustrate in the next example. Since the projector light is relatively bright (see section 3.3.3), exposure time is relatively small, about 10ms. In robotics, vision is usually used to gather global information about the scene, so the lens is more likely a wide angle lens than a zoom lens. Thus, assuming a camera with a relatively wide angle lens (e.g. principal distance at 1200 pixels), and the scene is at $1m$. Say d is the distance on the object corresponding to one pixel. Then $\frac{1pix}{1200pix} = \frac{d}{1m} \Rightarrow d \approx 0.8mm$.
  For the moving object to be in two pixels during the same camera integration time, it has to cross this distance in $10ms$, thus has to move at a speed of $> 8\frac{cm}{s}$. Motion blur of a single pixel will not influence our system however: section 5.4.2 will calculate the accuracy, and concludes that the contribution of one pixel error in the camera image is $\pm 1mm$. Consider an application that needs an accuracy of $\pm 1cm$, the average error is $0.5cm$: the application will start to fail starting from a 5 pixel error. Thus, for objects moving faster than $\pm 40\frac{cm}{s}$ at a distance of $1m$ deconvolution algorithms will need to be applied to compensate for motion blur. These however are not considered in this thesis and belong to future work.

- *Relative 6D position between camera and projector*: the rotational invariance ensures that whatever part of the pattern is visible in whatever orientation, it will always be interpreted correctly, independent of the relative orientation between camera and projector.

- *Out of focus is allowed*: for the chosen implementation, being out of focus is not a problem, as the centre of gravity of the detected blob will be used, which is not affected by blur. (this is the case for any symmetrical shape). Using the centre of gravity in the camera image is only an approximation. To be more precise, as Heikkilä [2000] states, the projection of the image plane of the centre of an ellipse is not equal to the centre of the ellipse projected in the image plane (due to the projective distortion). Correcting equations are presented in [Heikkilä, 2000]. Fortunately, robotics often uses wide angle lenses, and those have a larger depth of field than zoom lenses, so out-of-focus blurring will be less of a problem.

3.4 Pattern adaptation

As slide projectors were replaced by beamers in the nineties, patterns no longer had to be static. We use this advantage online in several ways: blob position, size and intensity adaptation. This way the sensor actively changes the circumstances to retrieve desired information, this is **active sensing**.

3.4.1 Blob position adaptation

Robot tasks do not need equidistant 3D information. To perform a task, certain regions have to be observed in more detail, and for certain regions a very coarse reconstruction is sufficient. Fortunately, with the proposed system, it is easy to redistribute the blobs anywhere in the projection image to sense more densely in one part of the scene, and less so in another.

3.4.2 Blob size adaptation

Making blobs smaller means less problems with depth discontinuities, and the possibility to increase the resolution. Making them larger signifies a more robust segmentation in the camera image. A **balance** between the two imposes itself. The factor one wants to control here is the size of the features in the camera image, through their size in the projector image.

Starting from a default resolution, the software resolves the correspondences. Then it is clear what part of the projector data has reached the camera. We adapt the pattern to the camera, as the other features are void anyway: the pattern is scaled and shifted to the desired location. Notice that we don't rotate the pattern: estimating the desired rotation would be a waste of processing power, as the pattern is recognisable at any rotation anyway. Hence, this is an adaptation in three dimensions: Let s be the scaling factor (range $0 \ldots 1$), w and h the image width and height in pixels, then the range of the horizontal shift is $0 \ldots (1-s)(w-1)$ and of the vertical shift $0 \ldots (1-s)(h-1)$.

The blob size in the projector image also needs to be reduced **as the robot moves** closer to its target: then during the motion the blob size in the camera image remains similar, but the corresponding 3D information becomes more dense and local.
A robot arm can also benefit from the extra degree of freedom of a **zoom camera**. When DCAM compliant *FireWire* cameras have this feature for example, it is one of the elements that can be controlled in the standardised software interface. Hence, the zoom can be adapted online by the robot control software, depending on what region is interesting for visual control.

3.4.3 Blob intensity adaptation

Section 4.2 explains how to calculate the responses of camera and projector to different intensities. Once these are known, and one knows which projector intensity illuminates a certain part of the camera image, then one can estimate

the **reflectance** of the material of that part. The only reason for this estimation would be to adapt the projector intensity on that patch accordingly: making sure that the transmitted (projected) features remain in the dynamic range of the receiver, the camera (see section 3.3.6). This is necessary, as under- or oversaturation do not produce valid measurements. It is not necessary to actually estimate the reflection coefficient, as every blob contains a part that is **near white** in the camera image: a part that has the projector intensity that corresponds to an almost maximum response of the corresponding camera image pixels. If it would be the maximum response, one could not detect oversaturation any more, as there would be no difference between an oversaturated output, and a maximal one.

Every blob in the image is adapted individually. One can simply adapt the intensity of both blob parts of each blob linearly according to the deviation of the part with the largest intensity from its expected *near* maximal output. This adaptation process does not necessarily run at the same frame rate as the correspondence solving runs, as for many applications the influence of reflectance variations is not such that it needs to be calculated at the same pace.

3.4.4 Patterns adapted to more scene knowledge

When one has more model knowledge about the scene (see section 6.3), other types of pattern adaptations are possible. Sections 8.3 and 8.4 of the experiments chapter contain two example of that type of active sensing: first a rough idea of the geometry of the pattern is formed using a sparse equidistant pattern, then details are sensed using a pattern that is adapted in position, size and shape according to this rough geometric estimation.

3.5 Conclusion

This chapter discussed the available choices one has in every step of the encoding pipeline. For every step, this chapter selects the choice that is most suited for the control of a robotic arm. First it discusses the code logic put into the pattern in section 3.2: it argues that a matrix of dots is a more interesting choice than a hexagonal dot organisation or a 2D grid pattern. Then the implementation of this code into an actual pattern in section 3.3 concludes that patterns with only grayscale features that allow to locally compare intensity differences are most broadly applicable. Finally section 3.4 explains how the pattern needs to be adapted online such that the sensor adapts itself to the robot, and not the other way around. For more detailed conclusions, see the conclusion paragraphs of sections 3.2 and 3.3.

Chapter 4

Calibrations

Great speakers are not born, they're trained.

Dale Carnegie

4.1 Introduction

This chapter identifies the communication channel: it estimates the parameters needed for the channel to function. In vision terms, this identification is called calibration. The parameters to estimate are:

- The **intensity responses** of camera and projector, as illustrated in figure 4.3 (see section 4.2).
- Parameters defining the geometry of the **light paths** in the camera and projector. These are called the intrinsic parameters (see section 4.3).
- Parameters that define the relative **pose** between camera and projector: these are the extrinsic parameters (see section 4.4).

Each of this sections will be introduced by explaining when/why the knowledge of these parameters are needed. Figure 4.1 places this section in the broader context of all processing steps in this thesis.

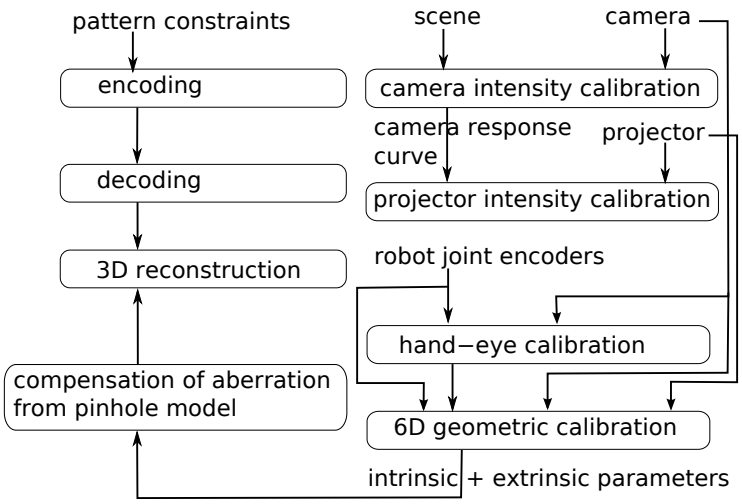


Figure 4.1: Overview of different processing steps in this thesis, with focus on calibration

4.2 Intensity calibration

Motivation: scene reflectance

Materials can reflect in a diffuse or a specular way. The directional diffuse reflection model is a combination of both, see figure 4.2. It produces a highlight, there where the angle of the incident light to the surface normal is equal to the viewing angle. This in combination with the different colours of the scene has a non-negligible result on the frequency and amount of reflected light. A solution to the complication of the specularities, is to identify and remove them in software, as in [Gröger et al., 2001]. The system has to be able to cope with different reflections because of colours and non-Lambertian surfaces.

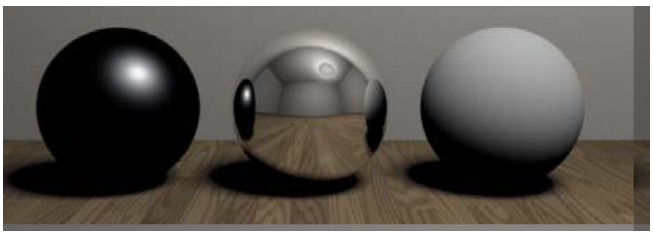


Figure 4.2: From left to right: directional diffuse, ideal specular and ideal diffuse reflection - by Cornell university

One wants to ensure that camera pixels do not over- or undersaturate to avoid clipping effects. On the other hand it is interesting to set the camera shutter speed such that the camera brightness values corresponding to the brightest

projector pixels are close to the oversaturation area: then the discerning capabilities are near maximal. In other words, the further the brightness values of the different codes are apart in the camera image, the better the signal to noise ratio.

Section 3.3.6 explains that the segmentation needs for the projected pattern are local **brightness (grayscale)** comparisons. Thus, since the pattern only uses a brightness ratio for each blob, one can do without explicitly estimating the surface reflectance for this decoding procedure. Then apply feedback to the brightness of each of the projected blobs such that the brightest of the two intensities in the blob follows the near maximum desired output. Figure 4.3 illustrates this feedback to the projector. The top left curve transforms the pixel values requested of the projector into projected intensities. The top right function accounts for the reflectance characteristics of the surface, for each of the blobs in the scene. The bottom right function then transforms the reflected light into camera pixel values. At this point, all of these curves are unknown.

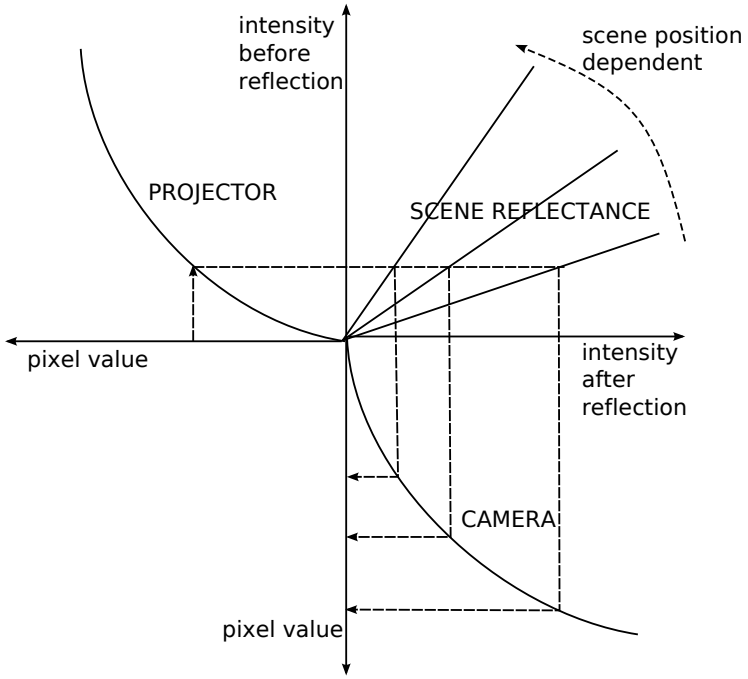


FIGURE 4.3: Monochrome projector-camera light model

Note that the model assumes the reflection curves to be linear. Linear indeed as the Phong reflection model, a combined reflection model with ambient, diffuse and specular reflections, has the form:

$$I_{reflected} \sim I_{ambient} + I_{incident}(\rho_{diffuse} \cos\theta + \rho_{specular} \cos\alpha^m) \qquad (4.1)$$

with $\rho_{diffuse}$ and $\rho_{specular}$ reflection constants dependent of the material, θ the angle between the incident light and the surface normal, and α the angle between the viewing direction and the reflected light. (m is the cosine fall-off, a shininess constant dependent of the material)
For this application, it is safe to assume $I_{ambient} << I_{incident}$. Hence attenuating the projector light $I_{incident}$ gives a linear decrease in the reflected light.

However, the feedback method of brightness adaptation may converge only slowly or suffer from overshoot, depending on the controller. One can also make the one time effort of estimating the response curves of both imaging devices. As correspondences are known, both camera and projector intensity values are also known for each blob. Then the only unknown element is the local reflectance characteristic (slope of the reflectance curve) at the location of each of the blobs. It is then easy to estimate the reflectance for each blob. This value can be used to make sure the camera remains in its dynamic range, by adapting the projector intensity accordingly.

Say the brightest part of a blob has an average projector brightness I_p and camera brightness I_c. Consider the response curves of camera and projector g_c and g_p respectively, where the horizontal axis contains the pixel value and the vertical axis the intensity. The scene locally has a reflection coefficient c (the slope of the reflectivity function). Then $c = \frac{g_c(I_c)}{g_p(I_p)}$. The desired projector output $I_{p,d}$ can then be calculated as

$$I_{p,d} = g_p^{-1}(\frac{g_c(I_{c,d})}{c})$$

This information can be used as feedforward, and can in combination with the intensity feedback deal with the different reflections.

Algorithm

This section presents the procedure used to identify the intensity response curves of camera and projector.

This work uses the recovery of high dynamic range radiance maps as presented by Debevec and Malik [1997]. Debevec and Malik do not specify which pixels have to be used as input to estimate the response curves. If one uses all available pixels for all F frames taken at different exposure settings, the system of equations to be solved becomes unnecessarily large. Ensure that even at the longest exposure setting, no pixel oversaturates. We choose those P pixels with well spread intensity values in smooth image areas, to avoid discretisation effects near edges. A $l \times l$ smoothing kernel detects the smooth parts of the image. ($l = 10$ here) The mask ensures selected points are not clustered together and are not at image edges. The algorithm is executed on the image with the largest exposure time, as its dynamic range, and thus the accuracy of the calculations, is higher than the other images. The scene used for this calibration can be any scene that has sufficient diversity in intensity values. To make sure this is the case, one can use a surface with an gradient from black to white.

Algorithm 4.1 Selection of pixels for intensity calibration: $pixSet \leftarrow \text{pixSelect}(P,\, l)$

$r \leftarrow \max_{u,v}(Img) - \min_{u,v}(Img)$

$step \leftarrow \dfrac{r}{P}$

$Img_{blurred} \leftarrow Img * \dfrac{1_{l\times l}}{l^2}$ {convolve with constant kernel}

$Img_{sharpener} \leftarrow Img - Img_{blurred}$

$$Img_{mask} \leftarrow \left[\begin{array}{c|c|c} \mathbf{1}_{\frac{l}{2}\times\frac{l}{2}} & \mathbf{1}_{\frac{l}{2}\times(\mathcal{W}-l)} & \mathbf{1}_{\frac{l}{2}\times\frac{l}{2}} \\ \hline \mathbf{1}_{(\mathcal{H}-l)\times\frac{l}{2}} & \mathbf{0} & \mathbf{1}_{(\mathcal{H}-l)\times\frac{l}{2}} \\ \hline \mathbf{1}_{\frac{l}{2}\times\frac{l}{2}} & \mathbf{1}_{\frac{l}{2}\times(\mathcal{W}-l)} & \mathbf{1}_{\frac{l}{2}\times\frac{l}{2}} \end{array}\right]$$

for $i \leftarrow 0$ to P **do**

$Img_{val} \leftarrow \max(0, Img - \min_{u,v}(Img) - i \cdot step)$

$Img_{penalty} \leftarrow Img_{val} + Img_{sharpener} + Img_{mask}$

$(u_m, v_m) \leftarrow \arg\min_{u,v} Img_{penalty}$

$pixSet \leftarrow pixSet \cup (u_m, v_m)$

$$Img_{mask} \leftarrow Img_{mask} + \left[\begin{array}{c|c|c} \mathbf{0}_{(u_m-l)\times(v_m-l)} & \mathbf{0}_{2l\times(v_m-l)} & \mathbf{0} \\ \hline \mathbf{0}_{(u_m-l)\times 2l} & \mathbf{1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right]$$

end for

Algorithm 4.1 presents the details (note that the image is monochrome). By subtracting the blurred image from the original one, one becomes an image that has higher intensity values where the spatial frequencies are higher. This image will become part of the mask, as higher frequencies need to be avoided. Img_{mask} is another part of the mask: it is 0, except around the image edges, as these edges need to be avoided (as the convolution is problematic there). Then for every discretisation value, construct the image Img_{val} that has as lowest values the values that correspond to that discretisation value. These values have to be selected preferably. The penalty image $Img_{penalty}$ consists of this image, the image that avoids the intensity edges, and the image that avoids the image edges. The image coordinates corresponding to the minimal value are saved, and an image avoiding the direct neighbourhood of this point is added to the mask.

Once the pixels have been chosen, the resulting mathematical problem in the Debevec and Malik method is an optimisation problem that can be solved by least squares: using SVD decomposition. The (quadratic) objective function is the sum of terms that fit the data points to the curve, and a weighted sum of second derivatives at those points, to smoothen the curve. As an image contains discrete data, these second derivatives are approximated by second order central differences.

The function value corresponding to each pixel value is unknown. Let β be the pixel depth, the number of bits per pixel (often $\beta = 8$). In order to choose the number of pixels P and the number of images F, compare the number of

Table 4.1: Number of unknowns and equations in the Debevec and Malik minimisation problem

	unknowns	equations
	2^β response curve function values 2^β smoothing weights P film irradiance values	PF data fit equations 2^β smoothing equations
total	$P + 2^{\beta+1}$	$PF + 2^\beta$

unknowns with the number of equations, see table 4.1.

Hence, in order to have a overdetermined system of equations, F and P have to be chosen such that $P(F-1) > 2^\beta$. This thesis chooses $P = 40$, $F = 10 \Rightarrow 40 \cdot 9 > 256$.

Results

Now the calibration of the intensity response of the camera is complete. The next step is to do the same for the projector. For this, we model the projector as an inverse camera. Different shutter settings are equivalent to the different brightness levels in the projector image. (inverse exposure) We can only observe the intensity of the projector output through the camera. Therefore, we need to take into account the camera response calculated in the previous paragraph. In order to minimise the influence of the different reflectance properties of the scene materials, let the projector light reflect on a white uniform diffuse surface. This thesis then uses the same algorithm as for the camera. This procedure is similar to the one described in [Koninckx et al., 2005], but one does not need to study the different colour channels separately here. Figure 4.4 shows the results of algorithm 4.1 for both camera and projector. The camera response function approximates a quadratic function with a negative 2nd order derivative. Thus, for a fixed increase in intensity in the darker range, the pixel value increases relatively more, and for brighter environments, the pixel value increases relatively little. This is a conscious strategy by imaging device manufacturers, to imitate the response function of the human eye, which is even stronger non-linear. Namely, it approximates a logarithmic response to brightness, differences in darker environments result in a large difference in stimulus, differences in bright environments do not increase the stimulus much.

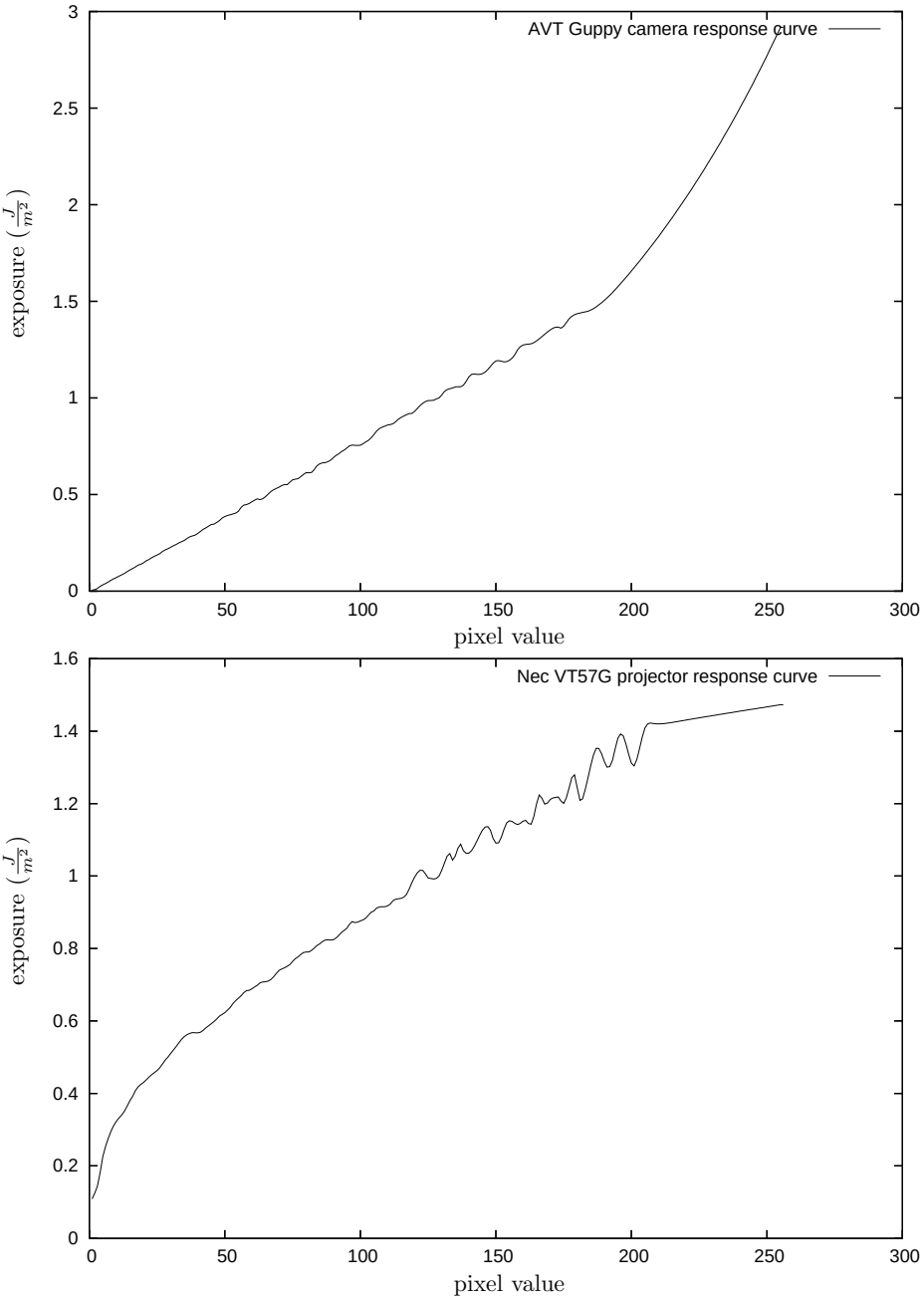


FIGURE 4.4: Camera and projector response curves

Vignetting

Another phenomenon that can be important in this type of calibration is the vignetting effect. Vignetting is an optical effect due to the dimensions of the lens: off-axis object points are confronted with a smaller aperture than a point on the optical axis. The result is a gradual darkening of pixels as they are further away from the image centre. Juang and Majumder [2007] perform an intensity calibration that includes the vignetting effect in its model. Apart from estimating the camera and projector response curves, he also estimates the 2D functions that define the vignetting effect (surfaces with the image coordinates as abscissas). Since this more general problem is higher in dimensionality, the solution is less evident: the optimisation procedure takes over half an hour of computing power.
This thesis chooses not to include such calibration here, as the effect is hardly noticeable for our setup: vignetting is dependent on the focus of the lens. As the focus approaches infinity, the effect becomes stronger: the camera iris and sensor are further apart. However, in our setup, we always focus on an object at a distance of $\pm 1m$. But more importantly, since the pattern elements used only rely on the **relative intensities** in each of the blobs, estimating this vignetting would not be useful. Not the vignetting in the camera image, nor the one in the projector image. It would not even be useful if the effect would be considerable: no global information is used in the segmentation, **only local comparisons** are made.

4.3 Camera and projector model

This section estimates the basic characteristics of the imaging devices. This is essentially the **opening angle** of the pyramidal shape through which they interact with the world, see the right hand side of figure 4.15. Clearly, this angle drastically changes the correlation between a pixel in the image of camera or projector and the corresponding 3D ray. It has thus also a **large influence on estimated location** of that feature in the 3D world. This is the case for all pixels, **except for the central pixel**: the ray through this pixel is not influenced by the intrinsic parameters. Thus, it is **under certain circumstances** possible to **experiment without** the knowledge of the **intrinsic parameters**. Consider for example a setup with only a camera and no projector, and a scene with only one object of interest. The paradigm of constraint based task specification [De Schutter et al., 2005] can then be used, to keep it in the centre of the camera image. The deviation from the image center is an error that can be regulated to 0 by adding this as a constraint. Then the robot knows the direction in which to move to approach the object, assuming that a hand-eye calibration was performed before. If the physical size of the object is known, comparing the sizes of the projection of the object in the camera image before and after the motion, results in the distance to the object. Clearly, the class of robot tasks that can be performed without camera calibration is limited, but one should remember to keep things simple by not calibrating the camera when it is not needed.

4.3.1 Common intrinsic parameters

These characteristics of the optical path are rather complex. Therefore, reduce this complexity by using a frequently used camera model: the pinhole model. The top drawing of figure 4.5 shows a schematic overview of a camera (here schematically with only one lens, although the optical path may be more complex). The focal length F is the distance (in meter) between the lens assembly and the imaging sensor (e.g. CCD or CMOS chip). We approximate this reality by the model depicted in the illustration on the lower half of figure 4.5: as if the object is viewed through a small hole. The principal distance f is the "distance" (in pixels) between the image plane and the pinhole. The orientation of the object is upside down. We can now rotate the image 180° around the pinhole: this leaves us with the model on the bottom of figure 4.5: this is the way the pinhole model is usually shown. This model contains some extra parameters to approximate reality better:

- (u_0, v_0) is the principal point (in pix): the centre of the image can be different from the centre of the pinhole model. This is caused by the imperfect alignment of the image sensor with the lens. Note that the origin of the axes u, v is in the centre of the image.
- the angle α between the axes u and v of the image plane.
- k_u and k_v are the magnifications respectively in the u and v directions (in pix/m).

These 5 parameters realise a better fit of the model, they are called the intrinsic parameters. We may choose to incorporate all or only part of them in the estimates or not, depending on the required modelling precision. The pinhole model linearises the camera properties. The intrinsic parameters are incorporated in the intrinsic matrices $\mathbf{K}_c$ and $\mathbf{K}_p$. Fk_u is replaced by one parameter f_u, as it is not useful to estimate the focal distance itself: the pinhole model does not need physical distances. $\frac{Fk_v}{\sin(\alpha)}$ is replaced by the principal distance f_v and $\frac{-Fk_u}{\tan(\alpha)}$ is estimated as the skew s_i:

$$\mathbf{K}_i = \begin{bmatrix} F_i k_{u,i} & \frac{-F_i k_{u,i}}{\tan(\alpha_i)} & u_{0,i} \\ 0 & \frac{F_i k_{v,i}}{\sin(\alpha_i)} & v_{0,i} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_{u,i} & s_i & u_{0,i} \\ 0 & f_{v,i} & v_{0,i} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

for $i = c, p$. Hence, there are 10 DOF in total. For the estimation of these intrinsic parameters, see the section about the 6D geometric calibration, section 4.4, as the intrinsic and extrinsic parameters are often estimated together.

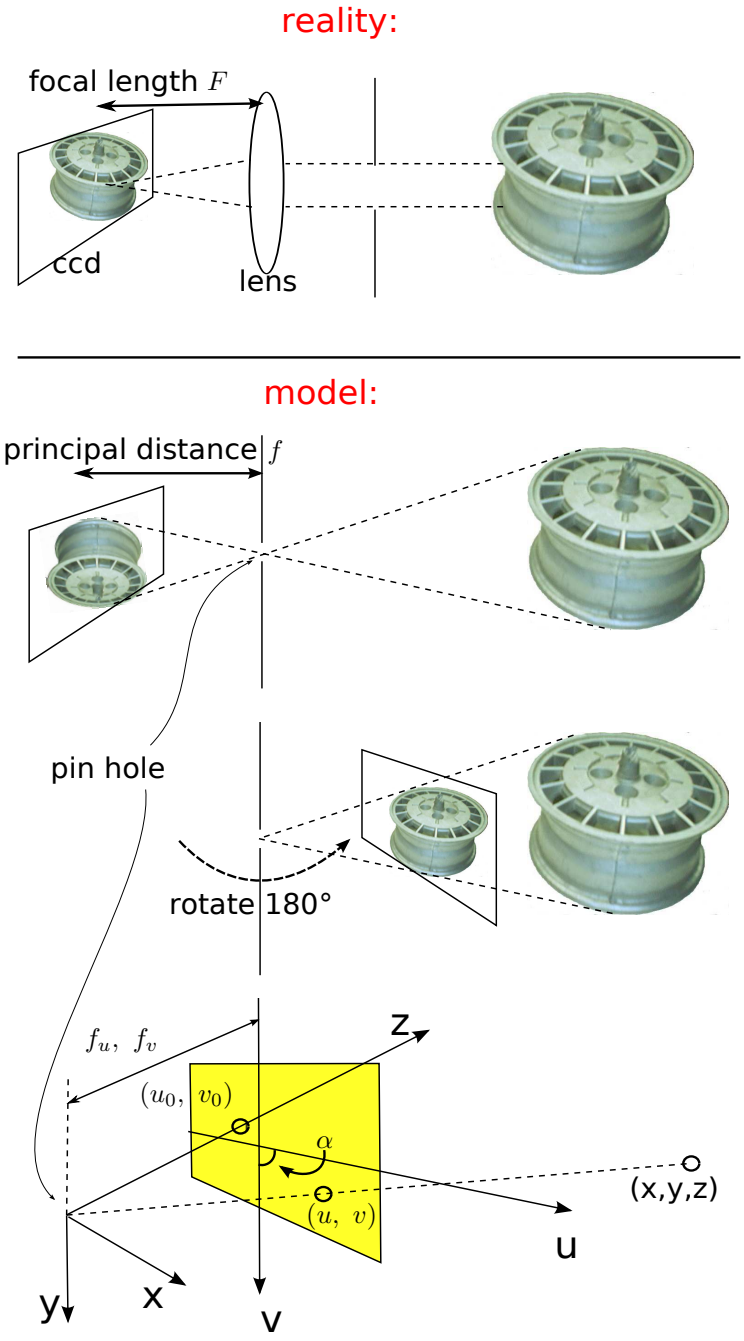


FIGURE 4.5: Pinhole model compared with reality

4.3.2 Projector model

Calibrating a projector is similar to calibrating a camera: it is also based on the pinhole model. One of the differences is that a projector does not have a symmetrical opening angle, as a camera does. The position of the LCD panel

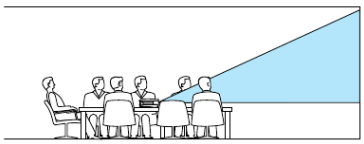


Figure 4.6: Upward projection

and the lenses is such that the projector projects upwards: only the upper part of the lens is used. This is of course a useful feature if the projector is to be used for presentations as in figure 4.6, where the projection screen is usually at the height of the projector and higher. However, for this application it is not useful: but one has to take this geometry into account.

An easy model to work with this asymmetry, is to calculate with a virtual projection screen that is larger than the actual projection screen [Koninckx, 2005]. The left side of figure 4.7 is a side view of the projector: it indicates the actual projection: angles β and γ. We expand the upper part (angle β) to a (virtual) symmetrical opening angle. Now one has a larger projection screen on which one only projects on the upper part in practice. The height above the central ray at a certain distance is called B, and the height below this ray is G. Let $\mathcal{H}_p$ be the actual height of the projector image, and $\mathcal{H}'_p$ be the virtual one, then:

$$\mathcal{H}'_p = \frac{2B}{B+G}\mathcal{H}_p = \frac{\sin(\beta)\cos(\gamma)}{\sin(\beta+\gamma)}\mathcal{H}_p$$

For example, in case of the NEC VT57 projector used in the experiments and shown in figure 4.7: $\mathcal{H}'_p = 1.67\mathcal{H}_p$. The right side of figure 4.7 shows a top view of the projector: seen from this angle, the opening angle is symmetrical both in reality and in the projector model.

Figure 4.8 summarises the models for camera and projector. In order not to overload the figure, it shows only one principal distance for each imaging device and no skew.

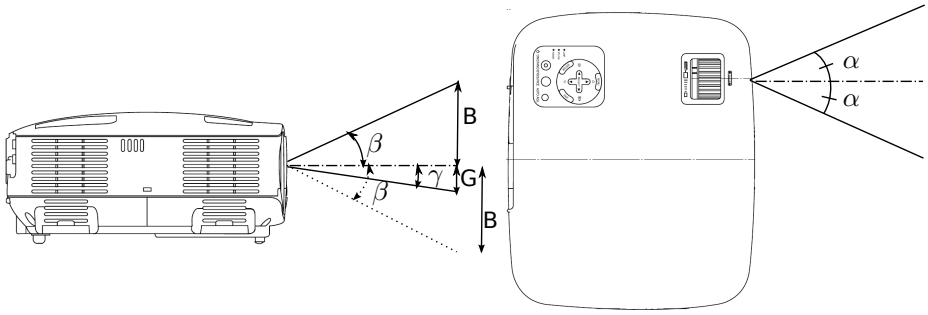


FIGURE 4.7: Asymmetric projector opening angle

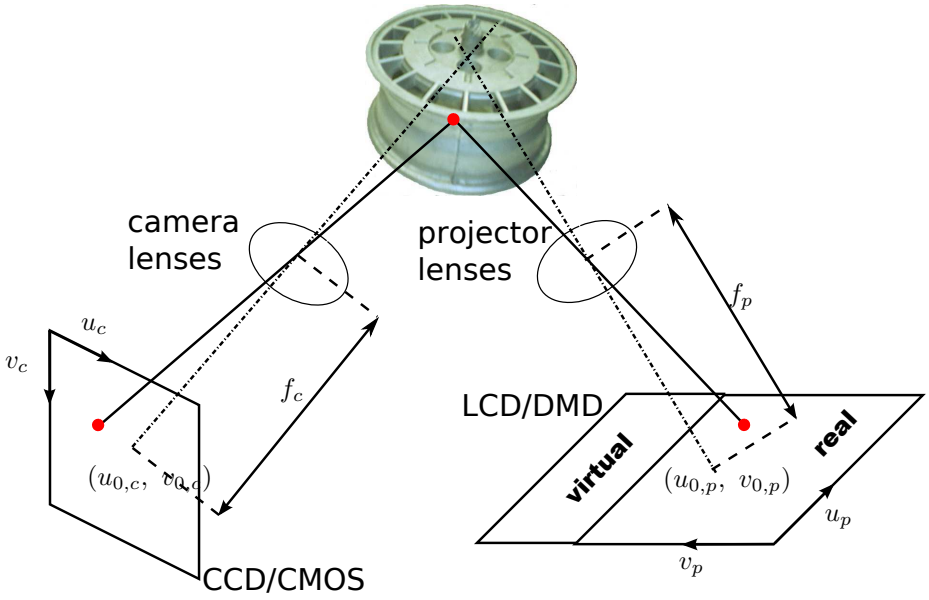


FIGURE 4.8: Pinhole models for camera - projector pair

4.3.3 Lens distortion compensation

In order to fit reality even better to the model, one needs to incorporate some of the typical lens-system properties in the model: lenses have different optical paths than pinholes. Incorporating this extra information does not require a completely different model: one can add the information that deviates from the pinhole model, on top of that model. Of all lens aberrations, we only correct for radial distortions, as these have a more important effect on the geometry of the image than other aberrations. To describe radial distortion, Brown [1971]

introduced the series:

$$\mathbf{u_u} = \mathbf{u_d} + (\mathbf{u_d} - \mathbf{u_0}) \sum_{i=1}^{\infty} \kappa_i ((u_d - u_0)^2 + (v_d - v_0)^2)^i$$

where $\mathbf{u_d} = (u_d, v_d)$ (distorted), $\mathbf{u_u} = (u_u, v_u)$ (undistorted) and $\mathbf{u_0} = (u_0, v_0)$ (principal point). As for every i the contribution of κ_i is much larger than the one of κ_{i+1}, usually only the first κ_i, or the first two κ_is are non-zero, yielding the polynomial approximation $r_u = r_d(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4)$ where $r_j = \sqrt{(u_j - u_0)^2 + (v_j - v_0)^2}$ for $j = u, d$.

Applying this technique would mean estimating two extra parameters, increasing the dimensionality of the calibration problem.
Radial distortion is an inherent property of every lens and not a lens imperfection: for a wide-angle lens (low focal length) the radial distortion is a barrel distortion (fish eye effect) that can be compensated for by positive κ_is. For tele lens (high focal length) it is a pincushion distortion: compensate this using negative κ_is. Pincushion distortion is only relevant for a focal length of $150mm$ or higher: a zoom this strong is not useful for the robotic applications studied in this thesis, where we work with objects at short range. Hence, one only needs to incorporate barrel distortion here.
Perš and Kovačič [2002] present an analytical alternative for barrel distortion based on the observation that the parts of the image near the edges (the more distorted parts) appear like they could have been taken using a camera with a smaller viewing angle that is tilted. Then distances in these parts appear shorter than they are due to the tilt. Straightforward geometric calculation based on this virtual tilted camera with smaller viewing angle result in an adapted pinhole model, with radial correction:

$$r_u = -\frac{f}{2} \frac{e^{-\frac{2r}{f}} - 1}{e^{-\frac{r}{f}}}$$

with f the principal distance. This model is only useful for cameras that are not optically or electronically corrected for barrel distortion, normal webcams or industrial cameras are not. Otherwise, identifying the κ_is is a good way of identifying the distortion compensation performance of a *smart camera*.

The projector model of section 4.3.2 is not only useful for the 3D calibration, but also for this lens distortion compensation. Radial distortion is defined with respect to an optical centre. The optical centre of the projector is not near $(\frac{\mathcal{W}_p}{2}, \frac{\mathcal{H}_p}{2})$ but rather near $(\frac{\mathcal{W}_p}{2}, \frac{B\mathcal{H}_p}{B+G})$.

Concluding, the compensation of radial distortion would normally introduce a non-linear parameter that needs to be estimated. But this can be avoided: this thesis does not introduce an extra dimension. We apply this compensation by Perš and Kovačič to both camera and projector. From this point on, the notation $\mathbf{u}$ will be used for $\mathbf{u_u}$, in order not to overload the subscript (other subscripts needed is one to indicate camera or projector, and one point index).

4.4 6D geometry: initial calibration

Initial calibration signifies the first geometric calibration of the setup. Since later on we intend to adapt the calibration parameters during motion, this is opposed to the next section: calibration tracking. If one wants to triangulate between camera and projector, these parameters need to be known. One could omit their estimation by using the less robust structure from motion variant: triangulation between different camera positions. Apart from the hand-eye calibration, these do not require the estimation of these geometric parameters.
Pollefeys [1999] and Hartley and Zisserman [2004] describe the different calibration techniques in more detail. Here we give only a short overview of relevant techniques. All these techniques have in common that they take the localisation (or tracking) of the visual features as an input. This is a prerequisite for calibration, as is the labelling described in chapter 5.3: in this section we can assume that the n_0 correspondences between camera and projector are known (at time step $t = 0$). Figure 4.11 for example shows two of these correspondences. Let the image coordinates in the projector image be $\mathbf{u}_{p,0,i}$, and the corresponding image coordinates in the camera image $\mathbf{u}_{c,0,i}$, with $i = 0..n_0$.

4.4.1 Introduction

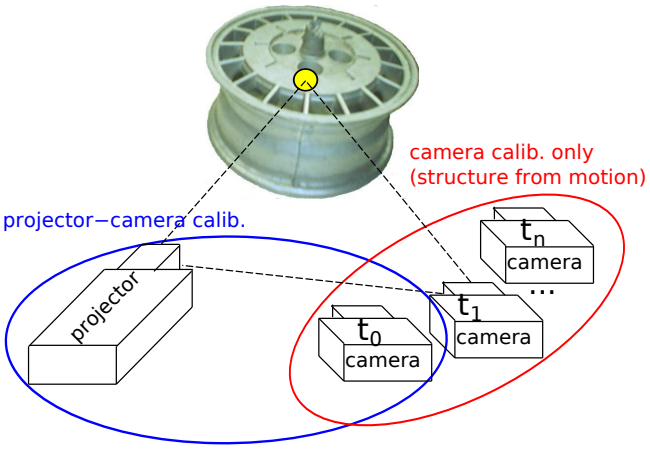


FIGURE 4.9: Calibration of extrinsic parameters between projector & camera (in space), or between two cameras (in time)

One could triangulate between a first camera position, a camera position later in time and the point of interest. Then we track image features between several poses of the camera: calculating the **optical flow**. Then one uses structure from motion to deduct the depth: the projector is only used as a feature generator to simplify the correspondence problem by making sure there are always sufficient features. It uses different viewpoints that are not separated in space but in time, see figure 4.9. Pagès et al. [2006] for example describes such a system.

But the **baseline**s in the optical flow calibration are typically **small**, and the larger the baseline the better the conditioning of the triangulation. Moreover, this restricts the system to static scenes: one would not be able to separate the motion of the camera and the motion of the scene. If one can also estimate the position of the projector, that extra information can be used to obtain a better conditioning, and the capability to work with dynamic scenes. Indeed, we can also base the triangulation on the baseline between projector and camera: a wider baseline. Pagès et al. [2006] perform no calibration between projector and camera. Only a rough estimation of the depth is used, objects that are far from planar are approximated as planar objects (with the same z value). This is possible as the IBVS Pagès et al. use is very robust against errors in the depth. However, he points out that, using the depths from a calibrated setup, better performance can be achieved.

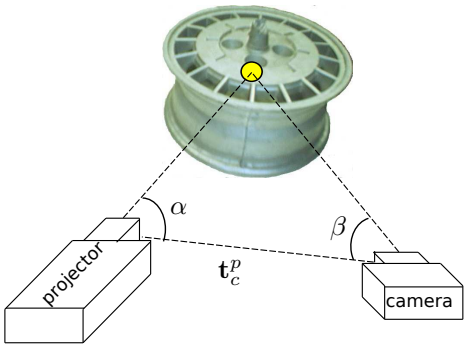


Figure 4.10: Angle-side-angle congruency

Thus, the system acquires the depths using triangulation between camera, projector and the point of interest. To calculate the height of a triangle, one needs information about some of its sides and angles. One can use the angle-side-angle congruency rule here as indicated in figure 4.10: the triangle is fully determined if the distance $|\mathbf{t}_c^p|$ between camera and projector and the angles α and β are known.

Of course we want to calculate the depth of several triangles at the same time. Figure 4.11 shows two of those triangles. It is essential to know which point belongs to which triangle: this is the correspondence problem, schematically indicated in figure 4.11 using differently coloured dots. The projector creates artificial features in order to simplify this correspondence problem considerably. The encoding chapter – chapter 3 – and the segmentation and labelling sections – sections 5.2 and 5.3 – explain how to keep the projected elements apart. In order to calculate $|\mathbf{t}_c^p|$, α and β three pieces of information are needed:

- the pixel coordinates of the crossing of each of the rays with their respective image plane (the correspondence problem).
- the 6D position of the frame $\{x_p, y_p, z_p\}$ with respect to the camera frame $\{x_c, y_c, z_c\}$.

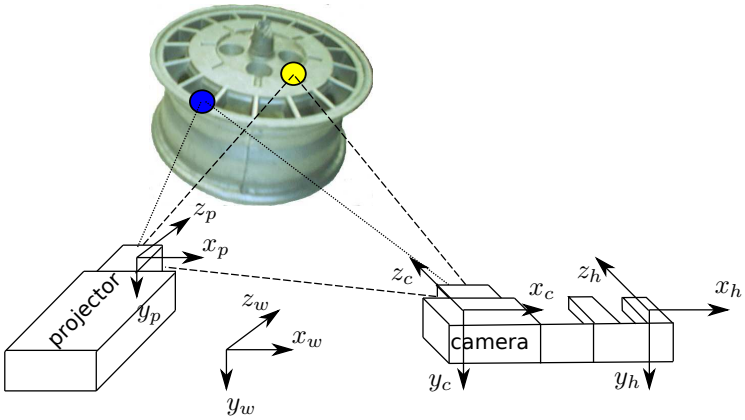


FIGURE 4.11: Frames involved in the triangulation

- The characteristics of the optical path in the camera and projector, to relate the previous two points.

In order to find the relationship between frames $\{x_p, y_p, z_p\}$ and $\{x_c, y_c, z_c\}$, one can make use of the hand-eye calibration that defines the relation between $\{x_c, y_c, z_c\}$ and $\{x_h, y_h, z_h\}$ (the h stands for "hand"), and the encoder values of the robot that give an estimation of the relation between $\{x_h, y_h, z_h\}$ and the world coordinate frame $\{x_w, y_w, z_w\}$.

Projection model

The robot needs to estimate 3D coordinates of points in the scene with respect to the world coordinate frame $\{x_w, y_w, z_w\}$, see figure 4.11. The matrix $\mathbf{R}_p^w$ represents the 3 rotational parameters of the transformation between world and projector coordinate frame: it is the rotation matrix from the world frame to the projector. Analogously $\mathbf{R}_c^p$ rotates from projector to camera frame. In a minimal representation, 6 of the 9 parameters in the matrices are redundant. This thesis uses Euler angles to represent the angles. Euler angles have singularities and can thus potentially lead to problems. At a certain combination of Euler angles – a singularity – a small change in orientation leads to a large change in Euler angle (this is where the same rotation is represented by several combinations of angles). Therefore one could change the parametrisation to the non-minimal representation with quaternions, or the minimal representation of an exponential map. For the Euler angle representation, this thesis uses the $z-x-z$ convention, with ϕ, θ and ψ the Euler angles of the rotation between projector and camera. The only singularity is then for $\theta = 0$: the z-axes of camera and projector are parallel. In that case, triangulation is anyhow not possible, so singularities in Euler angles will not be a problem in this case. Hence, assuming an Euler angle

representation:

$$\mathbf{R}_c^p = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{t}_p^w$ and $\mathbf{t}_c^p$ contain the corresponding translational parameters: three in each vector.

Of the 3 translational extrinsic parameters between camera and projector, one cannot be estimated using images only, since these images do not provide information on the size (in meter) of the environment. In other words, only two translational parameters are identifiable. Imagine an identical environment, but scaled, miniaturised for example: all data would be the same, so there is no way to tell that the length of the baseline has changed. One can only estimate this last parameter if the physical length of an element in the image is known. Hence, the reconstruction equations use a similarity sign instead of an equality sign. For $i \leftarrow p,\ c$, for j the index of the point:

$$\begin{aligned} \begin{bmatrix} u_{i,j} \\ v_{i,j} \\ 1 \end{bmatrix} &\sim \mathbf{K}_i \mathbf{R}_w^i \left(\begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} + \mathbf{t}_w^i \right) = \mathbf{K}_i \mathbf{R}_i^{wT} \left(\begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} - \mathbf{t}_i^w \right) \\ &= \mathbf{K}_i [\mathbf{R}_i^{wT} | -\mathbf{R}_i^{wT}\ \mathbf{t}_i^w] \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix} \equiv \mathbf{K}_i \begin{bmatrix} R_{11,i}^w & R_{12,i}^w & R_{13,i}^w & \tau_1 \\ R_{21,i}^w & R_{22,i}^w & R_{23,i}^w & \tau_2 \\ R_{31,i}^w & R_{32,i}^w & R_{33,i}^w & \tau_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \end{aligned}$$

where $u_{p,5}$ for example is the undistorted horizontal pixel coordinate of the 6th point in the projector image. $\mathbf{K}_i$ is defined by equation 4.2. And $\tau_k = (-\mathbf{R}_i^{wT}.\mathbf{t}_i^w)_k,\ k = 1..3$. Let the projection matrix $\mathbf{P}_i \equiv \mathbf{K}_i[\mathbf{R}_i^{wT} | -\mathbf{R}_i^{wT}.\mathbf{t}_i^w]$ for $i = p, c$:

$$\begin{aligned} \rho_{c,j} \begin{bmatrix} u_{c,j} \\ v_{c,j} \\ 1 \end{bmatrix} &= \mathbf{K}_c [\mathbf{R}_c^{pT} | -\mathbf{R}_c^{pT}\ \mathbf{t}_c^p] [\mathbf{R}_p^{wT} | -\mathbf{R}_p^{wT}\ \mathbf{t}_p^w] \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \\ \rho_{p,j} \begin{bmatrix} u_{p,j} \\ v_{p,j} \\ 1 \end{bmatrix} &= \mathbf{P}_p \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \end{aligned} \tag{4.3}$$

where $\rho_{i,j}$ is a non-zero scale factor: homogeneous vectors are equivalent under scaling, that is, any multiple of a homogeneous vector represents the same point in Cartesian space.

Implications of camera and projector positions

Section 3.2.2 described why the projector has a fixed position in this thesis, and the camera is moving rigidly with the end effector of a 6DOF robotic arm. Pagès et al. [2006] also use such a setup. The projector makes solving the correspondence problem easy, hence the baseline between different camera positions can be made relatively large, making a reasonable depth estimation from structure from motion possible.

Having a projector in a fixed position and a camera moving rigidly with the end effector implies a constantly changing baseline. Therefore we need to calibrate the 3D setup before the motion starts (explained in this section: section 4.4), and to update this calibration online as the relative position between camera and projector evolves (explained in section 4.5).

4.4.2 Uncalibrated reconstruction

In these systems, the intrinsic parameters, the ones that have been introduced in section 4.3, do not need to be estimated explicitly. Fofi et al. [2003] describes a camera-projector triangulation system to reconstruct a scene without estimating the extrinsic and intrinsic parameters. As we described in section 4.3, these parameters need to be known, and Fofi et al. uses that information but only implicitly. First he solves the correspondence problem, then the scene is reconstructed projectively. (for the different reconstruction strata, see [Pollefeys, 1999]) Then this projective reconstruction is upgraded to a euclidean one using several types of geometric constraints. Unfortunately, Fofi et al. cannot generate these equations automatically. Automating this constraint generation is described as future work, but has remained so ever since.

4.4.3 Using a calibration object

A possibility to estimate intrinsic and extrinsic parameters is using a calibration object. A calibration object is any object for which the correspondence problem can easily be solved. Hence it has clear visual features, of which the 3D coordinates are known with respect to a frame attached to the object itself. A planar object does not provide enough 3D information to calibrate the camera. The lower part of figure 4.13 shows an example of such object: two planar chess boards at right angles. After the chess board detector, one knows which 3D point corresponds to which image space point. It has been demonstrated [Dornaika and Garcia, 1997] that non-linear optimisation to estimate intrinsic and extrinsic parameters outperforms a linear approximation. Tsai [1987] estimates a subset of the parameters linearly, the others iteratively as non-linear parameters. Dornaika and Garcia [1997] perform a fully non-linear joint optimisation of intrinsic and extrinsic parameters.
Also the projector can be calibrated in this way, through the camera. The projector patterns are 1D Gray coded binary patterns, both in vertical and horizontal direction, see for example the left side of figure 4.12. On the right of this figure, a visual check: projector features at the chess board corners.

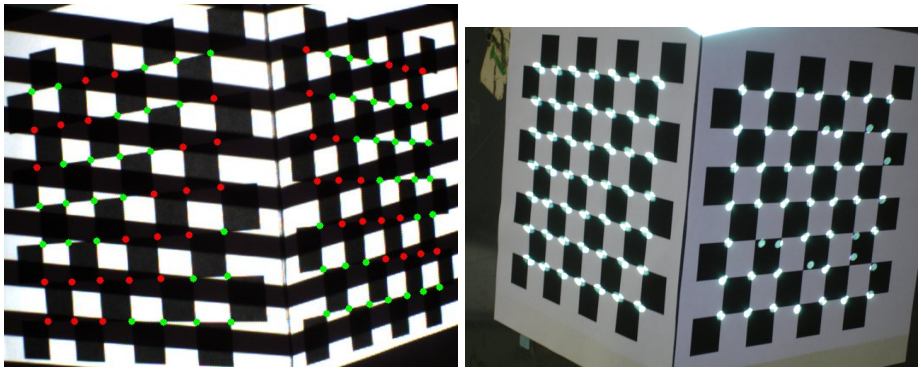


Figure 4.12: Calibration of camera and projector using a calibration object

Using 3D scene knowledge is a relatively straightforward technique, but there are several downsides to it:

- it is a batch technique. If the baseline changes during robot motion, one needs an incremental technique to update the calibration parameters. Self-calibration can be made incremental.
- the calibration objects needs to be constructed precisely: it is relatively unrobust against errors in the 3D point positions, the way these errors propagate through the algorithm is not well-behaved.
- it is easier if one could simply avoid having to construct and use a calibration object: it is a time consuming task.

Zhang [2000] improves this technique: he makes it less sensitive to errors, and removes the need for a non-planar object. Several viewpoints of the same known object (a chessboard for example) are sufficient.

4.4.4 Self-calibration

The term self-calibration indicates that one does not use 3D scene knowledge to estimate the intrinsic and extrinsic parameters but only images, see the upper half of figure 4.13. Here the image space correspondences from different points of view are known but not any 3D information. Compare this to the calibration with a known object, where one needs only image coordinates from a single view and the corresponding 3D Cartesian coordinates. **This work chooses self-calibration** for its most important setup (see figure1.1), because the disadvantages of calibration using a calibration object as discussed in the previous section, do not hold for self-calibration.
Self-calibration of a stereo setup is a broad subject, and deserves a more elaborate explanation as in [Pollefeys, 1999]. This section will only discuss a possible technique that is useful for the setup discussed in this thesis (see figure 1.1). A general discussion is beyond the scope of this thesis.

All of the following methods treat the calibration as an **optimisation problem**. This section reviews the advantages and disadvantages of some of the applicable techniques to make a motivated calibration choice at the end of the section.

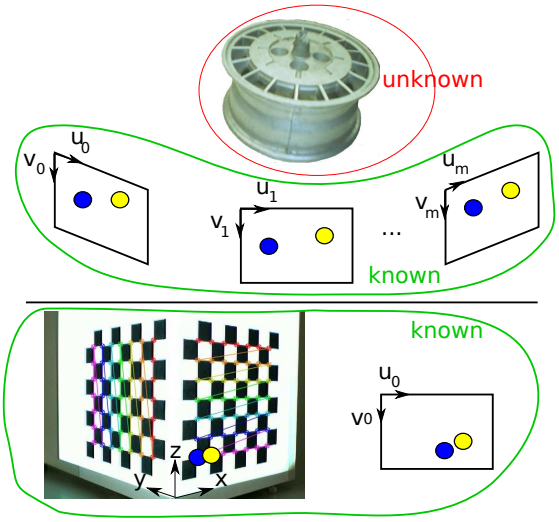


FIGURE 4.13: Top: self-calibration, bottom: calibration using calibration object

Optimisation in Euclidean space

Introduction In practice the half rays originating from the camera and the projector that correspond to the same 3D point on the scene, do not intersect but cross. This is among others due to discretisation errors, calibration errors and lens aberration. Unless more specific model knowledge is available, the point that is most likely to correspond to the physical point is the centre of the smallest

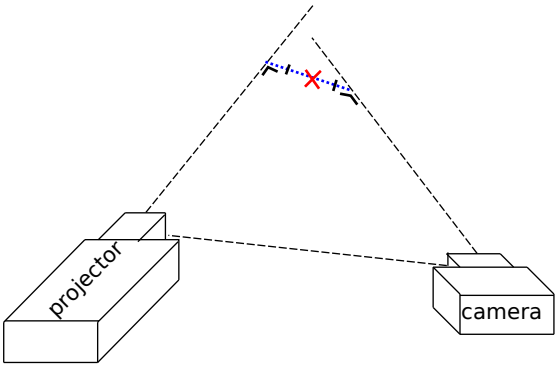


Figure 4.14: Crossing rays and reconstruction point

line segment that is perpendicular to both rays. The (red) cross in figure 4.14 indicates that point.

This approach retrieves the parameters by searching the minimum of a high dimensional cost function, in this sense it is a brute force approach. The cost function is defined as the sum of the minimal distances between the crossing rays: the sum of the lengths of all line segments like the (blue) dotted one of figure 4.14. The length of the baseline is added as an extra term to the cost function to ensure that the solution is not a baseline with length 0. The minimum of that function is the combination of parameters that best fit the available data, and hence the solution of this estimation problem.

This is a straightforward approach, with one big problem:
the **curse of dimensionality**. That is, the volume (search space) increases exponentially with the dimensions. So every dimension that does not need to be incorporated in the problem, simplifies the problem considerably. That is why in these approaches both camera and projector model are often simplified to a basic pinhole model with only one parameter: the principal distance. This is a reasonable approximation since the effect of an error in the principal distance on the result is much larger than the effect of an error in the principal point $(u_{0,i}, v_{0,i})$, the skew s_i or aspect ratio $(\frac{f_{u,i}}{f_{v,i}})$ for $i = c, p$. These methods assume the principal point to be in the centre of the image, the skew to be 0 and the aspect ratio to be 1. This reduces the number of **intrinsic parameters** for one imaging device from 5 to 1, or for both projector and camera, this is a **dimensionality reduction from 10 to 2**. These methods do not estimate radial distortion coefficients as this would increase the dimensionality again. They can afford to do so, as the effect of this distortion is also limited.
Since the entire setup can be scaled without effect on the images, the baseline (translation between the image devices) is not represented by a 3D vector, but by only 2 parameters of a spherical coordinate system: a zenith and azimuth angle. The rotation between the two devices is represented by the 3 Euler angles (the 3 parameters of the exponential map is also a good choice). Hence,

the dimensionality of the **extrinsic parameters** is **5 instead of 6**.

The setup contains two imaging devices so the total number of parameters is $n_e + 2n_i$, where n_e is the number of extrinsic parameters, and n_i the number of intrinsic parameters. When using a calibration object, the complexity of the model often incorporates two radial distortion coefficients and the 5D pinhole model as described in section 4.3. If we were to do the same here this problem is 19 dimensional, and hence prohibitively expensive to compute with standard optimisation techniques due to local extrema. In these techniques only the principal distance is used, so the problem is **7D** here.

Calibration If this 7D problem would have no local variables, it would be convex and its solution would be easy, even in a high-dimensional space. Unfortunately, the cost function is non-linear, and does have local minima, and hence due to the high dimensionality, finding its minimum is a computational problem. Furukawa and Kawasaki [2005] assumes the principal distance of the camera to be known, as it can be estimated using one of the techniques of section 4.4.3, for example the method of Zhang [2000]. Hence, Furukawa and Kawasaki do not propose a purely self-calibration method, a calibration grid is also involved. As the principal distance of the projector is harder to estimate, this parameter is kept in the cost function. He uses an iterative method to optimise this 6D cost function: the Gauss-Newton method. As this methods look attractive, this technique was reimplemented during this thesis. The input to the optimisation was simulated data, as shown in figure 4.15. On the right, some random 3D data points, and the pinhole models of camera and projector. On the left, the corresponding synthetic images, using a different colour for the projector and camera 2D image points.

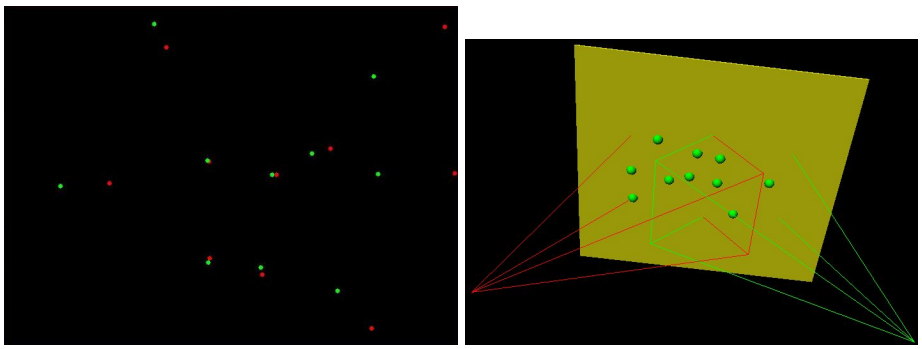


FIGURE 4.15: Calibration optimisation based on simulated data, according to Furukawa and Kawasaki

Thus, the cost function is dependent on 5 angles (three Euler angles, a zenith and a azimuth angle) and f_p. In order to use a gradient descent optimisation of this function, one needs the Jacobian of partial derivatives of the distances to each of the 6 parameters. If n is the number of distances between crossing rays, it is of size $n \times 6$, while the matrix of cost function values is of size $n \times 1$. As

the cost function contains 2-norms of vector, calculating it partial derivatives by hand is laborious: a symbolic toolbox calculates J once, and these expressions are hard coded in the optimisation program. Integrating the symbolic toolbox into the software, and delegating the substitution to the toolbox turns out to slow an already demanding computation down further. However, this approach does not necessarily converge for two reasons. One needs a good starting value to avoid divergence, and even then choosing the step size at each iteration can inhibit convergence. Furukawa and Kawasaki [2005] does not mention either one of these problems, it is unclear how they were able to avoid these difficulties. The experiments have shown that this approach is at least a very uncertain path. Indeed, figure 4.16 shows a 3D cut of the cost function: f_p and the three Euler angles are constant and have their correct values. Only the two parameters of the baseline change. On the right of the figure, one can see that some of the minima are slightly less deep than others: one can easily descend in one of the local minima using only a Gauss-Newton descent.

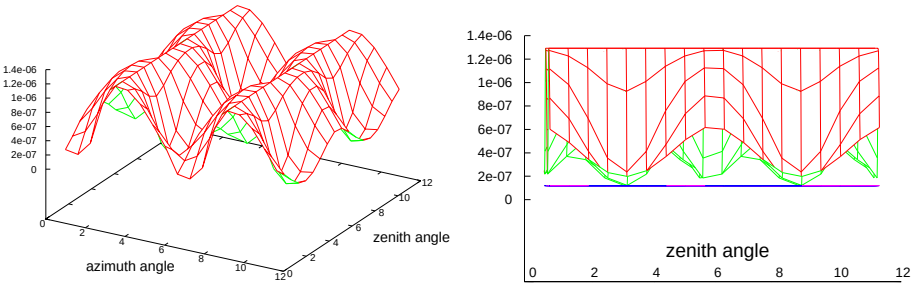


Figure 4.16: Cut of the Furukawa and Kawasaki cost function

Qian and Chellappa [2004] estimate the same parameters, but in a structure from motion context: the camera-projector pair is replaced by a moving camera. The two imaging devices are the same camera, but at different points in time. He optimises the same cost function, but with a **particle filter**. Hence, samples are drawn from this high dimensional function, and the most promising ones (the smallest ones), are propagated. If the initialisation of the filter uses equidistant particles, this approach is much more likely to avoid local minima than the previous one. The price to pay is a higher computational cost. Since it is a particle filter based procedure, no initial guess for the calibration parameters is required.
This procedure suffices to perform a self-calibration, but does not use any model knowledge. Indeed, for the setup of Qian and Chellappa, there is no additional model knowledge. However, in the context of a camera that is rigidly attached to the end effector, the robot encoders can produce a good estimate of the position of the camera. Using this knowledge in the self-calibration procedure will not only make it faster, but also more robust.

Optimisation using stratified 3D geometry: epipolar geometry

Introduction This paragraph first introduces some general notation and properties of epipolar geometry that will be used afterwards to estimate the calibration parameters.
Instead of perceiving the world in Euclidean 3D space, it may be more desirable to calculate with more restricted and thus simpler structures of projective geometry, or strata. Hence the word stratification. The simplest is the projective, then the affine, then the metric and finally the Euclidean structure. For a full discussion, see [Pollefeys, 1999].

Define $\mathbf{l}_{i,j}$ as the epipolar line for point $\mathbf{u}_j$, both for projector and camera ($i = p, c$, $j = 0 \dots n_0 - 1$): the line that passes through the epipole $\mathbf{e}_i$ and the projection $\mathbf{u}_{i,j}$ of $\mathbf{x}_j$ in image i. Let $\mathbf{u}_{i,j}$ and $\mathbf{e}_i$ be expressed in homogeneous coordinates: $\mathbf{u}_{i,j} = [u_{i,j}, v_{i,j}, 1]^T$, $\mathbf{e}_i = [u_{ei}, v_{ei}, 1]^T$. Then $\mathbf{l}_{i,j}[(l_{i,j,0}, l_{i,j,1}, l_{i,j,2}]^T$ is such that

$$l_{i,j,0}u_{ei} + l_{i,j,1}v_{ei} + l_{i,j,2} = l_{i,j,0}u_{i,j} + l_{i,j,1}v_{i,j} + l_{i,j,2} = 0 \Rightarrow \mathbf{l}_{i,j} = \mathbf{e}_i \times \mathbf{u}_{i,j}$$

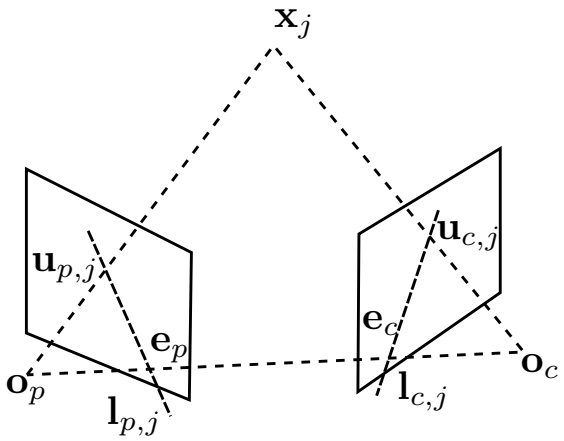


FIGURE 4.17: Epipolar geometry

For any vector $\mathbf{q}$, let $[\mathbf{p}]_\times \mathbf{q}$ be the matrix notation of the cross-product $\mathbf{p} \times \mathbf{q}$:

$$\mathbf{p} \times \mathbf{q} = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \equiv [\mathbf{p}]_\times \mathbf{q}$$

This matrix $[\mathbf{p}]_\times$ that is the representation of an arbitrary vector $\mathbf{p}$ with the cross product operator, is skew-symmetric ($[\mathbf{p}]_\times^T = -[\mathbf{p}]_\times$) and singular, see [Hartley and Zisserman, 2004].

Two matrices are relevant in this context:

- the **fundamental matrix F**: let $\mathbf{A}$ be the 3×3 matrix that maps the epipolar lines of one image onto the other: $\mathbf{l}_{c,j} = \mathbf{A}\mathbf{l}_{p,j}$. Then $\mathbf{l}_{c,j} = \mathbf{A}[\mathbf{e}_p]_\times \mathbf{u}_{p,j} \equiv \mathbf{F}\mathbf{u}_{p,j}$. A useful property is:

$$\mathbf{u}_{c,j}\mathbf{l}_{c,j} = 0 \Rightarrow \mathbf{u}_{c,j}^T \mathbf{F}\mathbf{u}_{p,j} = 0 \tag{4.4}$$

  As $\mathbf{A}$ is of rank 3 and $[\mathbf{e}_p]_\times$ is of rank 2, $\mathbf{F}$ is also of rank 2.

- the **essential matrix E**: let $\mathbf{x}_j$ be the coordinates of the 3D point in the world frame, $\mathbf{x}_{c,j}$ be the coordinates of that point in the camera frame, and $\mathbf{x}_{p,j}$ in the projector frame. Then according to section 4.4.1: $\mathbf{x}_{c,j} = \mathbf{R}_c^p(\mathbf{x}_{p,j} + \mathbf{t}_c^p)$. Taking the cross product with $\mathbf{R}_c^p\mathbf{t}_c^p$, followed by the dot product with $\mathbf{x}_{c,j}^T$ gives: $\mathbf{x}_{c,j}^T(\mathbf{R}_c^p\mathbf{t}_c^p \times \mathbf{R}_c^p)\mathbf{x}_{p,j} = 0$. This expresses that the vectors $\mathbf{x}_j - \mathbf{o}_c$, $\mathbf{x}_j - \mathbf{o}_p$ and $\mathbf{o}_p - \mathbf{o}_c$ are coplanar. Then:

$$\mathbf{E} \equiv [\mathbf{R}_c^p\mathbf{t}_c^p]_\times \mathbf{R}_c^p \Rightarrow \mathbf{x}_{c,j}^T \mathbf{E}\mathbf{x}_{p,j} = 0 \tag{4.5}$$

  $\mathbf{E}$ has 5 degrees of freedom: 3 due to the rotation, and 2 due to the translational (there is an overall scale ambiguity).
  The product of a skew-symmetric matrix and a rotation matrix has two equal singular values and the third is zero (and is thus of rank 2):

$$\forall \mathbf{B}_{3\times3}, \mathbf{R}_{3\times3} \text{ with } \mathbf{B}^T = -\mathbf{B}, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = I, |\mathbf{R}| = 1 : \tag{4.6}$$

$$\exists \mathbf{U}_{3\times3}, \mathbf{V}_{3\times3} : \mathbf{B}\mathbf{R} = \mathbf{U}\begin{bmatrix}\sigma & 0 & 0\\ 0 & \sigma & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{V}^T \tag{4.7}$$

  $\mathbf{E}$ for example is such a matrix, see [Huang and Faugeras, 1989] (or [Hartley and Zisserman, 2004]) for proof.

As $\mathbf{u}_{i,j} \sim \mathbf{K}_i\mathbf{x}_{i,j}$, equation 4.4 becomes: $(\mathbf{K}_c\mathbf{x}_{c,j})^T\mathbf{F}\mathbf{K}_p\mathbf{x}_{p,j} = 0$. Comparing with equation 4.5 results in

$$\mathbf{E} \equiv \mathbf{K}_c^T\mathbf{F}\mathbf{K}_p \tag{4.8}$$

Calibration This calibration approach divides the reconstruction into two parts: a projective and a Euclidean stratum. For a projective reconstruction, one only needs the fundamental matrix $\mathbf{F}$, which can be calculated based on only the correspondences. To do so, there are several fast direct and iterative methods like the RANSAC, 7- or 8-point algorithm, see [Hartley and Zisserman, 2004] for an overview. One can even incorporate the radial distortion, turning the homogeneous system of equations into an eigenvalue problem [Fitzgibbon, 2001]. This in case the analytical approach of Perš and Kovačič [2002], as presented in section 4.3.3, would not be sufficiently accurate.
The reconstruction can then be upgraded to a Euclidean one using identity 4.8, with $\mathbf{K}_p$ and $\mathbf{K}_c$ as unknowns. Now exploit property 4.6 in an alternative cost

function that minimises the difference between the first two singular values of $\mathbf{E}$. These are only a function of the intrinsic parameters: of the entries of $\mathbf{K}_p$ and $\mathbf{K}_c$. Hence the dimension of the optimisation problem has been reduced from $n_e + 2n_i$ (with n_e the number of extrinsic and n_i the number of intrinsic parameters) to $2n_i$. Mendonca and Cipolla [1999] choose to estimate two different principal distances, the skew and the principal point. So their problem is 10D, and solved by a quasi-Newton method: no second derivatives (Hessian) need to be known. The first derivatives (gradient) are approximated by finite differences. Gao and Radha [2004] use analytical differentiation by observing that the singular values of $\mathbf{E}$ correspond to the eigenvalues of $\mathbf{E}^T\mathbf{E}$. He applies this to a moving camera: hence using only one matrix of intrinsic parameters. Applying this technique to a camera-projector pair results in:

$$|\sigma\mathbf{I} - \mathbf{K}_p^T\mathbf{F}^T\mathbf{K}_c\mathbf{K}_c^T\mathbf{F}\mathbf{K}_p| = 0 \Rightarrow \sigma^3 + l_2\sigma^2 + l_1\sigma + l_0 = 0$$

with l_0, l_1 and l_2 function of the unknown parameters in the intrinsic matrices (Gao and Radha use 8 parameters: the skews are assumed to be 0). Applying property 4.6 reduces the cubic equation to a quadratic one with a determinant equal to 0: $\sigma^2 + 4l_1\sigma + l_1 = 0$. It can be written as a quartic polynomial, hence the derivatives can be calculated analytically and a Newton's method can be used.

The **dimensionality** of these optimisations can be **reduced** further. For example, if one assumes the skews to be 0 and the (principal distance) aspect ratio to be 1 and the principal point to be in the centre of the image, the problem is 2D instead of 10D. This is precisely what the optimisation methods in Euclidean space do: if this simplified model is useful for the application considered, it can also be used in these calibration methods. One could even split this optimisation up: apply this technique to a moving camera (moving the end effector), then the essential matrix is equal to $\mathbf{K}_c^T\mathbf{F}\mathbf{K}_c$ and the cost function is one dimensional for the most basic pinhole model. The estimation of $\mathbf{K}_c$ can then afterwards be used to estimate $\mathbf{K}_p$ by solving the problem for the projector-camera pair. The extrinsic parameters can then also be calculated based on the left hand side of formula 4.5:

$$\mathbf{E} = \sigma\mathbf{U}\underbrace{\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}_{\mathbf{C}}\underbrace{\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{D}}\mathbf{V}^T \Rightarrow [\mathbf{t}_c^p]_\times = \sigma\mathbf{U}\mathbf{C}\mathbf{U}^T,\ \mathbf{R}_c^p = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

or $\mathbf{R}_c^p = \mathbf{U}\mathbf{D}^T\mathbf{V}^T$: see [Nister, 2004] to resolve this ambiguity. However, the **disadvantage** of this technique is that the fundamental matrix has singularities.

F is undefined when:

- the object is **planar** [Hartley and Zisserman, 2004]. This is a problem as this thesis does not assume objects to be non-planar. One would like to be able to do visual control along a wall for example.
- there is **no translation** between the imaging devices. This is no problem when the calibration is between projector and camera, as above. However, when one would servo using only camera (end effector) positions, not involving the projector, this can be a problem. Indeed, as the robot arm moves closer to its target, it is likely to move slower, and the translation between two successive camera poses will have a negligible translation component.

Nister [2004] presents a method based on this epipolar geometry, that can however **avoid the planar degeneracy problem**. He estimates the extrinsic parameters **given the intrinsic parameters**. The intrinsic parameters can then for example be calibrated separately using a calibration object. The smooth transition between planar and non-planar cases is due to not calculating a projective reconstruction before a Euclidean one, but determining the essential matrix directly. **E** is namely not degenerate when the scene is planar. Nister argues that fixing the intrinsic parameters drastically increases the robustness of the entire calibration. The method to calculate **E** is based on the combination of equations 4.8 and 4.4: let $v_{i,j,k} = (\mathbf{K}_i^{-1}\mathbf{u}_{i,j})_k$ for $i = p, c$; $j = 0 \dots 4$; $k = 0 \dots 2$ (the k_{th} element of the vector)

$$\underbrace{\left.\begin{array}{c}\text{for}\\ i = p, c\\ j = 0 \dots 4\\ k = 0 \dots 2\end{array}\right\}}_{v_{i,j,k} = (\mathbf{K}_i^{-1}\mathbf{u}_{i,j})_k} \mathbf{q}_j = \begin{bmatrix} v_{p,j,0} & v_{c,j,0} \\ v_{p,j,1} & v_{c,j,0} \\ v_{p,j,2} & v_{c,j,0} \\ v_{p,j,0} & v_{c,j,1} \\ v_{p,j,1} & v_{c,j,1} \\ v_{p,j,2} & v_{c,j,1} \\ v_{p,j,0} & v_{c,j,2} \\ v_{p,j,1} & v_{c,j,2} \\ v_{p,j,2} & v_{c,j,2} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{q}_0^T \\ \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \mathbf{q}_3^T \\ \mathbf{q}_4^T \end{bmatrix} \begin{bmatrix} E_{00} \\ E_{01} \\ E_{02} \\ E_{10} \\ E_{11} \\ E_{12} \\ E_{20} \\ E_{21} \\ E_{22} \end{bmatrix} = 0$$

In the next steps of the process one needs to execute steps for which the numerical value of the intrinsic parameters is needed, Gauss-Jordan elimination for example. Therefore, this method cannot be used here, as $\mathbf{K}_p$ and $\mathbf{K}_c$ are considered unknown. Chen and Li [2003] present a similar system where the essential matrix is estimated directly considering all intrinsic parameters known, but applied to a structured light system. The following technique avoids the planarity difficulties while keeping the intrinsic parameters unknown.

Optimisation using stratified 3D geometry: virtual parallax

Introduction If the scene is planar, one can define a **homography** (or collineation) between the 3D points in the scene and the corresponding 3D location of the points in the image plane. It is a linear mapping between corresponding points in planes, represented by a 3×3 matrix $\mathbf{H}$. If a second camera is looking at the same scene, one can also calculate a homography between the points in the image plane of the first camera, and those of the second camera. These homographies define collineations in 3D **Euclidean space**, also often called homographies in the calibrated case. Thus for $\mathbf{x}_{p,j}$ a 3D point j with respect to the projector frame, and $\mathbf{x}_{c,j}$ the coordinate of the same point with respect to the camera frame: $\mathbf{x}_{p,j} = \mathbf{H}_p^c \mathbf{x}_{c,j}$.
One can also define a collineation between corresponding image coordinates in two image planes, in homogeneous coordinates. This is a homography in **projective space**, often called a homography for the uncalibrated case. It is represented by a 3×3 matrix $\mathbf{G}$, defined up to a scalar factor. Hence:

$$[u_{p,j} \ v_{p,j} \ 1]^T \equiv \mathbf{u}_{p,j} \sim \mathbf{G}_p^c \mathbf{u}_{c,j}$$

Despite the title of their paper, Li and Lu [2004] for example present not an uncalibrated, but a half self-calibrated, half traditionally calibrated method. Both the intrinsic and the extrinsic parameters of the projector are precalibrated using two types of calibration objects using techniques similar to the ones of section 4.4.3, they are assumed to remain unchanged. Thus if one projects a vertical stripe pattern, the 3D position of each of the stripe light planes is defined. The intrinsic and extrinsic parameters of the camera are then self-calibrated, based on the homographies between the camera image plane and each of the known stripe light planes. It is unclear how exactly the correspondence problem is solved in this work, clearly using other stripe patterns that are perpendicular to or at least intersection with the vertical stripes. This results in an iterative reconstruction algorithm, that is relatively sensitive to noise. This noise sensitivity is then diminished using a non-linear optimisation technique.

Calibration Zhang et al. [2007] self-calibrates the extrinsic parameters of a structured light setup, assuming a planar surface in the scene, and assuming all intrinsic parameters known. The scene needs to be planar to be able to base the calibration on an homography between the camera and projector image plane. Even if the homography is defined between these planes, and not between the camera plane and the stripe light planes as in [Li and Lu, 2004], the scene does not need to be planar to calibrate based on this homography. To that end choose 3 arbitrary points to define a (virtual) plane: hence the name virtual parallax (best choose these 3 points as far apart as possible to increase the plane conditioning). For this plane, matrix $\mathbf{G}_p^c$ can be calculated using standard techniques, see Malis and Chaumette [2000]. Then in the expression

$$\mathbf{H}_p^c = \mathbf{K}_c^{-1} \mathbf{G}_p^c \mathbf{K}_p \tag{4.9}$$

only the intrinsic parameters are unknown. Now let ${}_c\mathbf{n}$ be the normal of this plane with respect to the camera frame. Then it can be proved that $[{}_c\mathbf{n}]_\times \mathbf{H}_p^c$ complies with property 4.6: the first two singular values are equal, the third is 0. Thus one can define a cost function that minimises the difference between the first two singular values [Malis and Chipolla, 2000]. One can solve this optimisation problem as in [Mendonca and Cipolla, 1999]. Having estimated $\mathbf{H}_p^c$ as a result of this optimisation, the intrinsic parameters follow from equation 4.9. The extrinsic parameters can be calculated from the knowledge that

$$\mathbf{H}_p^c = \mathbf{R}_p^c + \frac{\mathbf{t}_p^c \; {}_c\mathbf{n}^T}{{}_c\mathbf{n}^T(\mathbf{o}_c - \mathbf{x}_j)} \tag{4.10}$$

where $\mathbf{x}_j$ is a point on the virtual plane (the denominator is the distance between the camera and the plane). For methods to extract $\mathbf{R}_p^c$ and $\mathbf{t}_p^c$ and resolve the geometric ambiguity, see [Malis and Chaumette, 2000]. Note the robustness of this estimation increases as the triangle between the three chosen points in both camera and projector image is larger. Hence, one needs to take radial distortion into account, as large triangles have corner points far away from the image centre. Moreover, in robotics often wide-angle lenses are used to have a good overview over the scene, which increases the radial distortion even more.
Note the similarity between the virtual parallax and the epipolar technique. Indeed, an uncalibrated homography $\mathbf{G}$ and the corresponding fundamental matrix $\mathbf{F}$ are related by $\mathbf{F}^T\mathbf{G} + \mathbf{G}^T\mathbf{F} = 0$. Equally for the calibrated case: $\mathbf{E}^T\mathbf{H} + \mathbf{H}^T\mathbf{E} = 0$. Or, combining equations 4.5 (left hand side) and 4.10: $\mathbf{E} = [\mathbf{t}]_\times \mathbf{H}$.

Optimisation adapted to eye-in-hand setup

Introduction The calibration we propose for this eye-in-hand setup (see figure 1.1), is based on the last type of self-calibration: using the virtual parallax paradigm. The difficulty with this technique is that it needs reasonable **starting values** for $\mathbf{K}_p$ and $\mathbf{K}_c$. Here is where one can use the extra model knowledge due to the fact that the camera is rigidly attached to the end effector, knowing its pose through the joint values. Note that the proposed calibration in this section has not been validated experimentally yet (see the paragraph *Calibration proposal* below).

Hand-eye calibration Hand-eye calibration indicates estimating the static pose between end effector and camera, based on a known end effector motion. This will be useful for the calibration tracking of section 4.5. A large number of papers have been written on this subject, an overview:

1. The initial hand-eye calibrations separately estimated rotational and translational components using a calibration object, thus propagating the rotational error onto the translation [Tsai, 1989].
2. Later methods [Horaud et al., 1995] avoid this problem by simultaneously estimating all parameters, avoid a calibration object, but end up with a non-linear optimisation problem requiring good starting values for the hand-eye pose and intrinsic parameters of the camera.
3. A third generation of methods use a linear algorithm for simultaneous computation of the rotational and translational parameters: Daniilidis [1999] for example uses SVD based on a dual quaternion representation. However, he needs a calibration object for estimating the camera pose: both the poses of the end effector and of the camera are needed as input.
4. The current generation of algorithms avoids the use of a calibration grid, and thus uses structure from motion [Andreff et al., 2001]. It is a linear method, that considers the intrinsic parameters of the camera known. Andreff et al. discusses how different combinations of rotations and translations excite different parameters. For example 3 pure translations leads linearly to the rotational parameters and the translational scale factor (see below). Two pure rotations with non parallel axes excite both rotational and translational parameters, but in a decoupled way. So after having estimated the rotational parameters and the scale factor through two translations, one can discard the rotational part of these equations, and solve for the translational vector, which is determined up to the scale factor just computed.

All methods can benefit from adding more end effector poses to increase the robustness. Note that not all end effector motions lead to unambiguous image data for self-calibration. Pure translations or rotations are an example of motions that do. Schmidt et al. [2004] propose an algorithm that selects the robot motion to excite the desired parameters optimally.

Pajdla and Hlavác [1998] describe the estimation of the rotational parameters in more detail. Namely, using 3 known observer translations, and two arbitrarily chosen scene points that are visible in all 4 views (call these viewpoints $c0$ to $c3$). These points can be stable texture features if these are present in the scene. Otherwise the projector can be used to project a sparse 2D pattern to artificially create the necessary features. In practice it is more convenient to use projector features here, as one will need the projected data to estimate $\mathbf{K}_p$ afterwards. Hence, project a very sparse (order of magnitude 5×5) pattern of the type proposed in section 3.3.6. Then using the system of equations 4.3:

$$\rho_{ci,j}\mathbf{u_{ci,j}} = \mathbf{K}_c\mathbf{R}_c^{wT}(\mathbf{x}_j - \mathbf{t}_{ci}^w) \text{ with } \mathbf{u_{ci,j}} = \begin{bmatrix} u_{ci,j} & v_{ci,j} & 1 \end{bmatrix}^T$$

for $i = 0 \ldots 3$, $j = 0, 1$. Subtracting the equations for $i = 1 \ldots 3$ from the one for $i = 0$:

$$\rho_{c0,j}\mathbf{u_{c0,j}} - \rho_{ci,j}\mathbf{u_{ci,j}} = \mathbf{K}_c\mathbf{R}_c^{wT}(\mathbf{t}_{ci}^w - \mathbf{t}_{c0}^w) \tag{4.11}$$

for $i = 1 \ldots 3$, $j = 0, 1$. Now, subtracting the equation for $j = 1$ from the one for $j = 0$:

$$\begin{bmatrix} u_{c0,0} & -u_{ci,0} & -u_{c0,1} & u_{ci,1} \\ v_{c0,0} & -v_{ci,0} & -v_{c0,1} & v_{ci,1} \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \rho_{c0,0} \\ \rho_{ci,0} \\ \rho_{c0,1} \\ \rho_{ci,1} \end{bmatrix} = 0 \tag{4.12}$$

The reconstruction is only determined up to a scale, thus choose one of the ρ values, for example $\rho_{c0,0}$. Then the homogeneous system of equation 4.12 is exactly determined. The ρ values can then be used in equation 4.11: then all elements of this equation are known, except for $\mathbf{K}_c\mathbf{R}_c^{wT}$. A QR decomposition then results in both $\mathbf{K}_c$ and $\mathbf{R}_w^c$.

Starting value for camera intrinsic parameters The algorithm by Pajdla and Hlavác also results in the intrinsic parameters of the camera. This $\mathbf{K}_c$ can be used as a good starting value for the optimisation problem using the virtual parallax paradigm, see the algorithm below.

Starting value for projector intrinsic parameters The method by Pajdla and Hlavác [1998] can make a Euclidean reconstruction of any point in the scene using the equation

$$\forall j : \mathbf{x}_j = \rho_{c0,j}(\mathbf{K}_c\mathbf{R}_c^{wT})^{-1}\mathbf{u}_{c0,j} + \mathbf{t}_{c0}^w \tag{4.13}$$

Note that the scale factors $\rho_{i,j}$ of equation 4.3 are also known here: the physical dimensions of the robot are known, and this resolves the reconstruction scale ambiguity, as explained in section 4.4.1.
Reconstruct all observable projected points from one of the camera viewpoints, for example $c0$, and use this 3D data to estimate the intrinsic parameters of the projector. One can use the technique "with a calibration object" here (see section 4.4.3), as of these points both 2D and 3D coordinates are known. Indeed, the 2D coordinates are the projector image coordinates, and the 3D coordinates

are given in the world frame. One has no knowledge of the position of the projector yet, but that poses no problem, as the 3D coordinates do not need to be expressed in the projector frame, but the techniques in section 4.4.3 allow them to be expressed in *any* frame. Note that this algorithm does not excel in robustness. Therefore, if the results are insufficiently accurate, it is better to start with intrinsic parameters that are typical for the projector (see below for the case of a planar scene).

Increased robustness through multiple views Malis and Chaumette [2000] propose to extend the virtual parallax method to multiple view geometry (more than two viewpoints). This is done by composing the homography matrices of the relations between all these views, into larger matrices, both in projective space as in Euclidean space. Apply this technique on the camera-projector pair, where the end effector has executed m translations. Thus for the corresponding $m+1$ camera views, these matrices are of size $3(m+2)\times 3(m+2)$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{G}_p^{c0} & \dots & \mathbf{G}_p^{cm} \\ \mathbf{G}_{c0}^{p} & \mathbf{I}_3 & \dots & \mathbf{G}_{c0}^{cm} \\ \vdots & & \ddots & \vdots \\ \mathbf{G}_{cm}^{p} & \mathbf{G}_{cm}^{c0} & \dots & \mathbf{I}_3 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} \mathbf{H}_p^{p} & \mathbf{H}_p^{c0} & \dots & \mathbf{H}_p^{cm} \\ \mathbf{H}_{c0}^{p} & \mathbf{H}_{c0}^{c0} & \dots & \mathbf{H}_{c0}^{cm} \\ \vdots & & \ddots & \vdots \\ \mathbf{H}_{cm}^{p} & \mathbf{H}_{cm}^{c0} & \dots & \mathbf{H}_{cm}^{cm} \end{bmatrix}$$

In this setup (see figure 1.1), applying only the two view virtual parallel method would be discarding useful information available in the other views. One can easily gather this information from supplementary views, as the robot arm already needs to perform three translations to estimate a starting value for $\mathbf{K}_c$. The associated algorithms are similar, but the robustness of calculating with the super-collineation $\mathbf{G}$ and the super-homography $\mathbf{H}$ is larger.

We extend the technique described in Malis and Cipolla [2000] (as suggested in the paper) for the case where a virtual plane is chosen for each viewpoint, as these points are chosen based on the projected features. It would pose an unnecessary extra constraint on the chosen points if they have to be visible from all viewpoint.

The image coordinates $\mathbf{u}_{i,k}$ of a point with index k in image i are related to the image coordinates $\mathbf{u}_{j,k}$ of that point in image j: $\mathbf{u}_{i,k} \sim \mathbf{G}_{ij}\mathbf{u}_{j,k}$ (for 1 projector image and $m+1$ camera images). Out of these equations for $i = 0 \dots m, j = 0 \dots m$, one extracts an estimate for the super-collineation $\hat{\mathbf{G}}$. It can be proved that $\mathbf{G}$ is of rank 3, and thus has 3 nonzero eigenvalues, the others are null. Imposing this constraint improves the estimate $\hat{\mathbf{G}}$. Malis and Cipolla [2000] describes an iterative procedure to derive new estimates based on the previous estimate and the rank constraints.

The super-homography matrix $\mathbf{H}$ for a camera-projector pair is given by:

$$\mathbf{H} = \mathbf{K}^{-1}\mathbf{G}\mathbf{K} \text{ where } \mathbf{K} = \begin{bmatrix} \mathbf{K}_p & \mathbf{0}_{3\times3} & \dots & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{K}_c & \dots & \mathbf{0}_{3\times3} \\ \vdots & & \ddots & \vdots \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \dots & \mathbf{K}_c \end{bmatrix}$$

Since $\mathbf{G}$ is of rank 3 and $\mathbf{K}$ is of full rank, $\mathbf{H}$ is also of rank 3. After normalisation (see Malis and Cipolla [2000]), $\mathbf{H}$ can be decomposed as $\mathbf{H} = \mathbf{R} + \mathbf{T}$ where:

$$\mathbf{T} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{t}_p^{c0}\ {}_{c0}\mathbf{m}_{c0} & \mathbf{t}_p^{c1}\ {}_{c1}\mathbf{m}_{c1} & \dots & \mathbf{t}_p^{cm}\ {}_{cm}\mathbf{m}_{cm} \\ \mathbf{t}_{c0}^{p}\ {}_{p}\mathbf{m}_{c0} & \mathbf{0}_{3\times3} & \mathbf{t}_{c0}^{c1}\ {}_{c1}\mathbf{m}_{c1} & \dots & \mathbf{t}_{c0}^{cm}\ {}_{cm}\mathbf{m}_{cm} \\ \mathbf{t}_{c1}^{p}\ {}_{p}\mathbf{m}_{c1} & \mathbf{t}_{c1}^{c0}\ {}_{c0}\mathbf{m}_{c1} & \mathbf{0}_{3\times3} & \dots & \mathbf{t}_{c1}^{cm}\ {}_{cm}\mathbf{m}_{cm} \\ \vdots & & & \ddots & \vdots \\ \mathbf{t}_{cm}^{p}\ {}_{p}\mathbf{m}_{cm} & \mathbf{t}_{cm}^{c0}\ {}_{c0}\mathbf{m}_{cm} & \mathbf{t}_{cm}^{c1}\ {}_{c1}\mathbf{m}_{cm} & \dots & \mathbf{0}_{3\times3} \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{R}_p^{c0} & \dots & \mathbf{R}_p^{cm} \\ \mathbf{R}_{c0}^{p} & \mathbf{I}_3 & \dots & \mathbf{R}_{c0}^{cm} \\ \vdots & & \ddots & \vdots \\ \mathbf{R}_{cm}^{p} & \mathbf{R}_{cm}^{c0} & \dots & \mathbf{I}_3 \end{bmatrix}, \text{ with } {}_i\mathbf{m}_j = \frac{{}_i\mathbf{n}_j{}^T}{{}_i\mathbf{n}_j^T(\mathbf{o}_i - \mathbf{x}_j)}$$

with $\mathbf{x}_j$ is any point on the virtual plane for the camera (or projector) pose j, and ${}_i\mathbf{n}_j$ is the normal of the virtual plane corresponding to camera (or projector) pose j expressed in the frame of that camera (or projector) pose i. For a technique to extract the rotational and translational components from $\mathbf{H}$, see Malis and Cipolla [2000]. Now this matrix is known, one can optimise the cost function:

$$\mathcal{C} = \sum_{i=-1}^{m} \sum_{j=-1}^{m} \frac{\sigma_{i,j,1} - \sigma_{i,j,2}}{\sigma_{i,j,1}} \tag{4.14}$$

where $\sigma_{i,j,k}$ is the kth singular value of $[\mathbf{n}_{ij}]_\times \mathbf{H}_{ij}$. Indeed, this matrix has two equal singular values, and one 0 singular values (as explained above). Every independent $\mathbf{H}_i^j$ provides the algorithm with two constraints (see Malis and Cipolla [2000]). With $m+1$ camera images with m end effector translations in between, one has $m+2$ images (including the projector image). This results in $m+1$ independent homographies, and thus $2(m+1)$ constraints to solve for at most $2(m+1)$ parameters. Therefore one needs at least four translations of the end effector to estimate all parameters: this results in 5 camera images and 1 projector image, and thus in 5 independent homographies. This is sufficient to fix the 10 DOF of the intrinsic parameters of the camera-projector pair.
Malis and Cipolla [2000] also presents a curve showing the noise reduction as the number of images increases: experiments have shown that at ≈ 20 images, the slope of the noise reduction is still considerable. Is is therefore interesting to execute more than these four translation. However, each new image implies a new end effector movement and thus more calibration time. As a balance, choose $m = 10$ for instance.

Calibration proposal We propose a calibration that is bootstrapped from a **structure from motion** calibration. In other words, starting from a calibration that does not involve the projector pose, see section 4.3, exploiting the known camera motion :

1. **Project** a sparse 2D pattern ($\approx 5 \times 5$) as in section 3.3.6.
2. For $i = 0 \ldots 3$
   - Save the camera and projector coordinates of all **blobs** that can be **decoded** in image i.
   - Choose 3 visible projected features in this image such that the area of the corresponding triangles in camera and projector image is large. These points define a virtual **plane** for this end effector pose, store their image coordinates in the projector and camera images.
   - **Translate** to the next end effector pose. Stop the end effector after the translation during at least one camera sensor integration cycle, to avoid (camera) **motion blur**. Another reason to stop the translation is that the joint encoders are faster at delivering their data than the camera is (needs an integration cycle): otherwise there would be timing complications.

   Of all decodable blobs of the first item, discard all projector and camera image coordinates, except for 2 points that remain visible in all 4 of these end effector poses i.
3. Calculate an initial estimate for $\mathbf{K}_c$, using the algorithm by Pajdla and Hlavác [1998] (see above). This way one uses the knowledge that is in the known transformations between the camera poses.
4. Calculate an initial estimate for $\mathbf{K}_p$, using the calibration technique with known 3D coordinates (see above), or — if the accuracy is insufficient — simply use an estimate typical for the projector (see below: what in case of a planar scene).
5. **Identify the 6D pose between camera and projector**. To that end:
   - Use the two view correspondences to estimate the (uncalibrated) collineation $\mathbf{G}_p^{c0}$ [Malis and Chaumette, 2000].
   - Estimate the **normal** to the plane associated with camera pose $c0$. To that end, estimate the depth of the 3 points that determine the plane according to equation 4.13.
   - Calculate $\mathbf{H}_p^{c0}$ using equation 4.9 (for $c = c0$), with the estimates from items 3 and 4.
   - Determine the right hand side of equation 4.10 in terms of the rotation and translation between camera and projector (use the value for the normal from the previous step).
6. Perform a **hand-eye calibration**: use the technique by Andreff et al. [2001] as presented above. As input at least three images with two pure rotations in between to estimate $\mathbf{t}_c^{ee}$ up to a scale factor. The information of step 3, the algorithm by Pajdla and Hlavác [1998], results in the scale factor and $\mathbf{R}_c^{ee}$.

7. The remainder of the steps are **optional** to iteratively improve the result. For $i = 4 \dots m$ execute points 2 and 3 of step 2: find 3 decodable points for every image i, that form relatively large triangles in projector and camera image, then move the end effector. If all end effector motions are pure translations, $\mathbf{R}$ is filled with $\mathbf{I}_3$ blocks, except for $\mathbf{R}_{ij} = \mathbf{R}_p^{c0}$ for $i = 0$, $j = 1..m+1$, and $\mathbf{R}_{ij} = \mathbf{R}_p^{c0T}$ for $i = 1..m+1$, $j = 0$.

8. Calculate the super-collineation (as explained above).

9. Minimise the cost function 4.14 for the intrinsic parameters, and reintroduce the updated parameters in $\mathbf{H}$ as $\mathbf{K}^{-1}\mathbf{G}\mathbf{K}$. Decompose this newly calculated $\mathbf{H}$ into improved plane normals and 6D poses. This leads to a new cost function 4.14 that can be minimised again, etc.

The **complexity** of this procedure is linear (virtual parallax, estimation of $\mathbf{K}_p$ and $\mathbf{K}_c$), except for the last step. This non-linear optimisation is however an optional step, as first estimates have already been calculated before. Malis and Cipolla [2002] suggests that the method can be improved using a probabilistic noise model. Indeed, for example particle filtering seems a good choice for this last optional step.

The procedure above needs to be slightly adapted when the scene is **planar** for two reasons:

- Malis and Chaumette [2000] explain how estimation of the uncalibrated collineation is different when the scene is planar compared with when it is not (in the latter case one has to choose a virtual plane, by definition of the collineation).

- Step 5 of the above procedure needs a non-planar scene: a planar scene does not excite all parameters sufficiently. If the scene is planar, replace this step by a estimating a matrix $\mathbf{K}_p$ based on the projector image size and distance. $\mathbf{K}_p$ is allowed to be coarsely calibrated in this step, as it is only a starting value for the optimisation afterwards. If d is the physical width of the projected image, then the estimated principal distance is $\frac{\mathcal{W}_p z_p}{d}$. For example for $\mathcal{W}_p = 1024pix$, $d = 0.5m$, $z_p = 1m$: $f_p = 2048pix$.

Kanazawa and Kanatani [1997] propose a planarity test to check is which case the scene at hand is. It is based on the homography $\mathbf{H}_p^c$ between the image coordinates in both imaging devices in homogeneous coordinates. The test is based on the property that, if the scene is planar $[u_p v_p 1]^T \times \mathbf{H}_p^c [u_c v_c 1]^T = 0$.
Note the duality: $\mathbf{G}$ is undefined for a non-planar scene, $\mathbf{F}$ for a planar scene. A more complex alternative to the virtual parallax method would be to switch model between the epipolar and homographic model, depending on the planarity of the scene: according to geometric robust information criteria [Konouchine and Gaganov, 2005].

4.5 6D geometry: calibration tracking

During the motion of the end effector part of the calibrated parameters changes, but in a structured way, such that one does not need to repeatedly execute the preceding self-calibration online.

- *Intrinsic parameters.* Unless a zoom camera is used, the intrinsic parameters of camera and projector remain constant. In those cases, these parameters remain constant. For zoom cameras (see section 3.4) one can approximate the relation between the zoom value as transmitted in software, and the intrinsic parameters. Note that not only the focal length is then subject to change, but also the principal point.

- *Extrinsic parameters.* The evolution of the extrinsic parameters is connected to the motion of the end effector through the hand-eye calibration (the latter remains static as well). Thus, forward robot kinematics allow to adapt the extrinsic parameters using the joint encoder values. This is the **prediction** step of the calibration tracking, or in other words a feedforward stimulus. As the robot calibration, encoder values and hand-eye calibration are all imperfectly estimated, a **correction** step is necessary to improve the tracking result. This feedback stimulus is based on camera measurements: the Euclidean optimisation technique by Furukawa and Kawasaki [2005], discussed in section 4.4.4 is useful here, but adapted to only incorporate the extrinsic parameters: consider the projector principal distance known. It needs good starting values, but these are available after the prediction step. The technique is based on Gauss-Newton optimisation, in order to keep the computational cost low, a few iterations suffice.

  Better still is to exploit the sparse structure in the optimisation problem, and to apply **bundle adjustment** [Triggs et al., 2000]. Bundle adjustment jointly optimises the calibration parameters by reducing the number of parameters in the optimisation problem, by making the problem more dense and easier to solve. It uses a statistical approach to improve the accuracy.

  Whatever the technique used is, this feedback can run at a lower frequency than the feedforward, or in other words at a lower OS priority. If the frequencies differ, one needs to incorporate the latest feedback correction in the feedforward adaptations that are not followed by a feedback step.

4.6 Conclusion

This section estimated a number of parameters that are essential to the 3D reconstruction of the projected blobs. These are parameters that characterise the pose between the imaging devices and the properties of these devices: modelling the lens assembly and the sensor sensitivity. These parameters can be estimated automatically in a short offline step before the robot task. There is no need to construct calibration objects. The calibration is based on certain movements of the robot that excite the unknown parameters, and the projection of a number of patterns: different intensities for the intensity calibration and a sparse 2D pattern for the geometric reconstruction and the lens assembly properties.
After this initial calibration, a few of the calibrated parameters need to be updated online (section 4.5). Now this knowledge will be used to reconstruct the scene in section 5.4.

Chapter 5

Decoding

The problem with communication ... is the illusion *that it has been accomplished.*

George Bernard Shaw

The projector and the camera communicate through structured light. This chapter treats all aspects of the decoding of the communication code the camera receives. Four aspects are important:

- The analysis of the camera images, discussed in section 5.2. This is comparable to reading individual letters, or hearing a collection of phonemes.
- Labelling, determining the coherence between the individually decoded features, discussed in section 5.3. Compare this to, because they are in a certain order, combining letters to words and sentences, or hearing sentences by combining a sequence of phonemes.
- Having knowledge about some relevant parameters of the environment, discussed in section 4. This is comparable to the information we have learnt to map the image of our left and right eye together, and not perceive two shifted images.
- Using the correspondences between to images to reconstruct the scene in 3D, as discussed in section 5.4, as humans can perceive depth.

Figure 5.1 places this section in the broader context of all processing steps in this thesis.

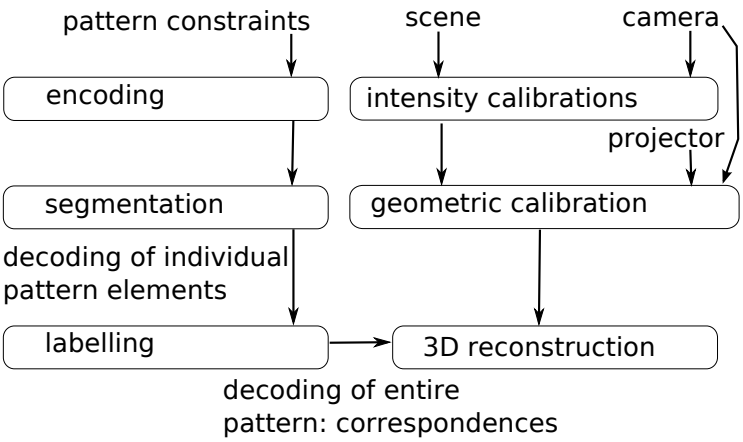


Figure 5.1: Overview of different processing steps in this thesis, with focus on decoding

5.1 Introduction

In order to decode the pattern successfully, the following assumptions are made:

- The surface lit by every of the elements of figure 3.19, is locally planar.
- That same surface has a constant reflection.

If the reality is a reasonable approximation of this model, the corresponding feature will be decoded correctly. If one or both of these assumptions are far from true, the corresponding feature will not be decoded, but this has no effect on the adjacent features (see the section of pattern logic: section 3.2).

5.2 Segmentation

This section decodes the individual elements of the projected pattern. The input of that process is a camera image, the output the codes of every element in the pattern.

5.2.1 Feature detection

The aim of this section is to robustly find the blobs in the camera image that correspond to the projected features. Figure 5.2 shows such a structured light frame to be segmented, captured by the camera. First the pixels of the background are separated from the foreground (see § Background segmentation), then the algorithm extracts the contours around those pixels (see § Blob detection). As a last step we check whether the detected blobs correspond to the model of the

projected blobs (see § Model control).
If the available processing power is insufficient for this procedure, constructing an image pyramid is a good idea to accelerate the segmentation, thereby gracefully degrading the results as one processes ever lower resolution images. (see also the segmentation of the spatial frequency case in section 3.3.5)

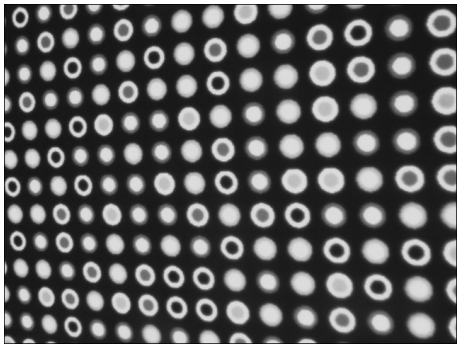


FIGURE 5.2: Camera image of a pattern of concentric circles

Background segmentation

The projector used for the experiments has an output of 1500 *lumen* (see section 3.3.3). This projector's brightness output is on the lower end of the 2007 consumer market (and hence so is its price), but it is able to focus an image at the closest distance available in that market: at about half a meter, which is interesting for robotics applications: we never need it to produce an image at more than a few meters. In a typical reconstruction application, the surface of the workspace on which light is projected is $\pm 1m^2$, thus the illumination is $\pm 1500\ lux$. Compare this with e.g. a brightly lit office, which has an illumination of $\pm 400\ lux$, or a TV studio: $\pm 1000\ lux$. Direct sunlight of course is much brighter: $\pm 50000\ lux$. So unless sunlight directly hits the scene, it is save to assume that any ambient light has an illumination that is considerably lower than the projected features. This is also clear in experiments: at the camera exposure time with which the projected features are not under- nor oversaturated, the background is completely black. Even though there often are features on the background with an ambient illumination that are visible by the human eye. The disadvantage is that one cannot do other vision processing on the image that uses natural features. But it can also be used to our advantage: a dark background is easy to separate background from foreground pixels. The algorithm:

- Find the pixel with the lowest intensity in the image
- Perform a floodfill segmentation using that pixel as a seed.

- If that floods a considerable part of the image, finish: we found the background, the result is a binary image.
- Otherwise the seed is inside a projected feature of which the interior circle is black: take the pixel with the 2nd (and if necessary afterwards 3rd, 4th...) lowest intensity value not in the direct neighbourhood of the previous pixel. Repeat the floodfill segmentation on the original image until a large fraction of the image is floodfilled.

Note that the algorithm is not dependent on any threshold (there is no need to tune a threshold that would depend on illumination conditions). This is the case although floodfill segmentation as such is dependent on parameters. But as the illumination difference between the underilluminated background and the projected features is large, a (low) threshold can be chosen such that the system remains functional independent of the reflectivity of the scene.

Blob detection

Starting from the binary image of the previous step, the Suzuki-Abe algorithm [Suzuki and Abe, 1985] finds the contours, just like for the segmentation of the spatial frequency case (see section 3.3.5). Discard the blobs with a surface that is much larger or smaller than the majority of the features. (these can for example be due to incident sunlight)

Model control

The system uses the knowledge it has about the projector pattern to help segment the camera image. It fits what it expects to see on what it actually sees in the camera image, and thus uses as much of the model knowledge as possible to reduce the risk of doing further processing on false positives: blobs that did not originate from the projector.

Assume that the surface of the scene is locally planar. This is a reasonable assumption, since the projected features are small. Then the projected circles are transformed into ellipses. Hence we fit ellipses to each of the blobs: those of which the fit quality is too low are discarded. This leads to an alternative to this segmentation: to use a generalised Hough transform, adapted to detect ellipses.

5.2.2 Feature decoding

In this section we decode the intensity content of each of the features, based on their location as extracted in section 5.2.1. As in the previous section, the algorithms do not depend on thresholds that need to be tuned to illumination and scene conditions. There's no free lunch: the price we pay for this is the increased computational cost of the algorithms. There are two phases in this feature decoding: the clustering of intensities to separate the inner circle and the outer ring, and the data association between the camera and projector intensities.

Intensity clustering

First the two intensities in each of the blobs are separated. We do not know what (absolute) intensities to expect, since in a first reconstruction the reflectance of the scene cannot be estimated yet. Because the algorithm needs to be parameter independent, data-driven only, it executes this intensity clustering procedure before thresholding to automatically find a statistically sound threshold.
For that, it uses **EM segmentation**. EM stands for Expectation Maximisation, a statistics-based iterative algorithm, or more precisely collection of algorithms, see [Dempster et al., 1977]. Tomasi [2005] describes the variant used here.

If one would make a histogram of the intensity values of all pixels in one blob, two distinct peaks would be visible. This histogram is made for each of the blobs. EM models the histogram data as Gaussian PDFs: it assumes a Gaussian mixture model. Hence we estimate not only the location of the mean, but also the variance of the data. The intensity where the probabilities of both PDFs are equal (where the Gaussians cross) defines the segmentation threshold. The input a EM algorithm needs, is the number of Gaussian PDFs in the Gaussian mixture model, and reasonable starting values for their means and standard deviations. The number of Gaussians is always two here, and for the second input we use the following heuristic:

- The starting value for the first mean is the abscissa corresponding to the maximum of the histogram.
- The projector uses four intensities in this case, this splits the corresponding histogram in three parts, see figure 5.3. If we model the falloff of two of the Gaussian PDFs in the middle between their maxima to be 95%, the horizontal distance to each of the maxima is 3σ, and thus $\sigma = \frac{255}{3 \cdot 2 \cdot 3}$ is the starting value for all standard deviations.
- Discarding all histogram values closer than 3σ to the first mean, take as starting value for the second mean the maximum of all remaining histogram values.

Performing a few EM iterations on actual data yields a result like in figure 5.4. If the data is not near the minimum or maximum of the intensity range, a Gaussian model fits well. However, near the edges the peaks logically shift away

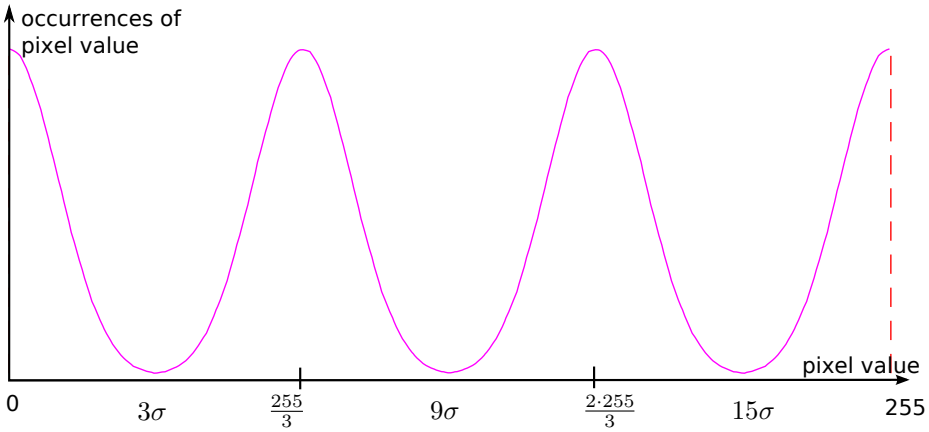


Figure 5.3: Standard deviation starting value

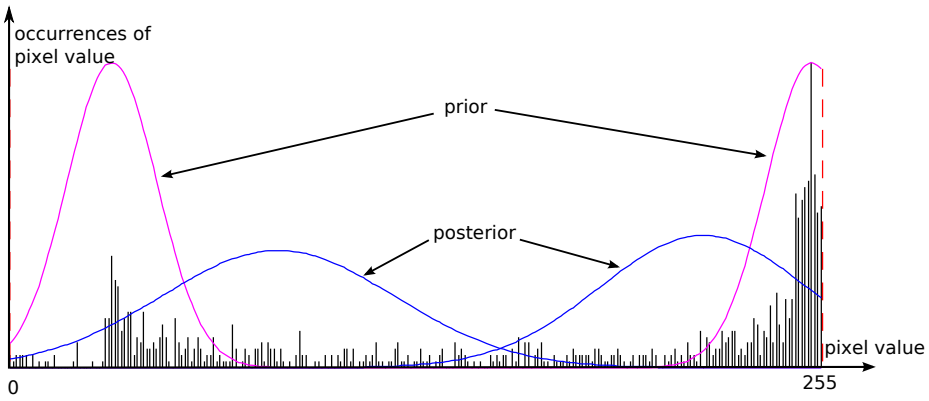


Figure 5.4: Automatic thresholding without data circularity

from the minimum and maximum intensity values. Therefore, make the data circular if a peak is near ($< 3\sigma$) the extrema of the intensity range, as can be seen on the left and on the right of figure 5.5. Data is added such that the histogram values for pixel values larger than 255 or smaller than 0 are equal to the histogram values for the intensity at equal distance to the nearest histogram peak. Let I be the pixel value. This way, the values around the histogram are extrapolated symmetrically around the peaks with $255 - I < 3\sigma$ or $I < 3\sigma$.

The dashed lines on the left and the right of figure 5.5 indicate the beginning and the ending of the actual histogram.
The two higher peaks in the figure are the initial estimates, and the broader PDFs are the result after a few EM steps: the maxima remain closer to their original maxima with this data circularity correction. The crossing of the PDFs

after EM determine the segmentation threshold.

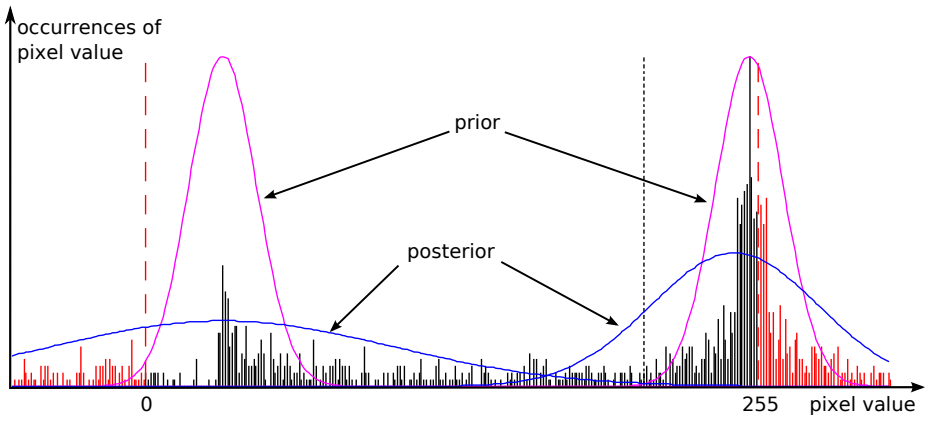


FIGURE 5.5: Automatic thresholding with mirroring

In this case, the initial estimates for the multivariate PDF would have segmented this blob fine, without EM. But consider for example the histogram in figure 5.6 (the red solid line). It has a non-Gaussian leftmost peak, and a right peak that has about a Gaussian distribution. The initial estimates for the means are at the peaks: indicated with a dotted line. Since the threshold is determined by the crossing of the two Gaussians, the threshold based on the prior only is in the middle between the prior Gaussians. The EM steps take into account that most of the data of the leftmost peak is to the right of that maximum: the mean shifts to the right. Hence, the posterior threshold also shifts to the right. A few EM steps suffice.

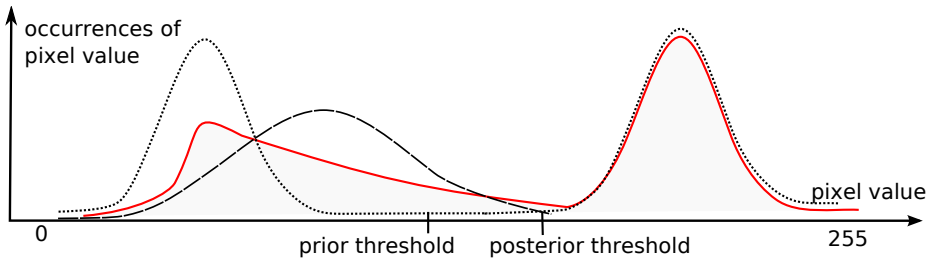


FIGURE 5.6: Difference between prior and posterior threshold

The dimensionality of the EM algorithm is only 1D, but if the available processing power would still be insufficient, one could choose for a different strategy in this processing step that returns a good approximation, considering the cost reduction: **P-tile segmentation**. Since about half of the pixels should have one intensity and the other half another, integrate the histogram values until

half of the pixels is reached: the corresponding intensity value is the threshold. In figure 5.5, a dotted line between the two dotted lines indicating the extra simulated data indicate this threshold: it is indeed near the threshold determined by EM segmentation. Optical crosstalk and blooming (see section 3.3.3) make P-tile segmentation less robust: the pixel surface of both intensities of a blob is the same in the projector image, but not in the camera image: the brighter part will appear larger than the darker part.
Both methods are global segmentation approaches. This section did not discuss local ones here, such as seeded segmentation, since the chosen seed might be inside a local disturbance in the scene. In that case the grown region will not correspond to the entire blob but only to a small part of it.

To increase robustness, one can check whether in the resulting segmentation, the one class of pixels is near the centre of the blob, and the other near the outer rim. Also the ratio between the inner and outer radii should remain similar to the one in the projector image: it is only changed by optical crosstalk and blooming. If either criterion is not satisfied, the blob needs to be rejected.

Intensity data association

The next step is to identify what projector intensities the means of the mixture model correspond to. Since each blob contains two intensities, the state space of this model is different: 2D instead of 1D.

Since one cannot assume the reflectance of the scene to be constant, one cannot use the absolute values of the means of the blobs directly. However, as one of them always corresponds to full projector intensity, we can express the other one as a percentage of this full intensity. This assumes that the reflected light is linear in the amount of incident light. This is physically sound: both standard diffuse and specular lighting models are, if one can neglect the ambient light: see equation 4.1.

Section 4.2 explains how the intensity response curves of camera and projector can be measured. Knowing these curves, one can estimate the actual intensity sent by the projector and the actual light received by the camera from the corresponding pixel values. Koninckx et al. [2005] proposes a similar system.
Assuming reflectance to be locally constant within a blob, the intensity ratio that reaches the camera is the same ratio as emitted by the projector. In the computations, one can scale both intensities linearly until the brightest one is 255, and then directly use a prior PDF, independent of the data. The number of Gaussians in this multivariate model is known: equal to the size of the alphabet, $a = 5$ in this case. In other words, let μ_{in} be the means of the pixel values of the pixels in the inner circle of the blob, and μ_{out} be the means of the pixel values of the outer ring pixels. Then define $\mathbf{I}$ as the couple of scaled intensity means such that if $\mu_{out} > \mu_{in} \Rightarrow \mathbf{I} = (\frac{255\mu_{in}}{\mu_{out}}, 255)$, otherwise $\mathbf{I} = (255, \frac{255\mu_{out}}{\mu_{in}})$.

Figure 5.7 displays this prior PDF according to the pattern implementa-

tion chosen in figure 3.19: maxima at $\boldsymbol{\mu_0}= (0, 255)$, $\boldsymbol{\mu_1}= (\frac{255}{3}, 255)$, $\boldsymbol{\mu_2}= (\frac{2 \cdot 255}{3}, 255)$, $\boldsymbol{\mu_3}= (255, \frac{2 \cdot 255}{3})$ and $\boldsymbol{\mu_4}= (255, \frac{255}{3})$. The standard deviations are such that the intersection between the peaks i and j for which $|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}|$ is minimal is at 3σ: $\sigma = \frac{255}{18} \approx 14$ with $\boldsymbol{\sigma}= (\sigma, \sigma)$. This prior distribution displays the probability for $\mathbf{I}$ to be equal to a certain couple $\mathbf{i}$: $P(\mathbf{I} = \mathbf{i})$. Since the distinction between the stochastic variable $\mathbf{I}$ and the particular value $\mathbf{i}$ is clear, the uppercase symbol can be omitted.

$$P(\mathbf{i}) = \sum_{i=0}^{a-1} \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\sigma}) \tag{5.1}$$

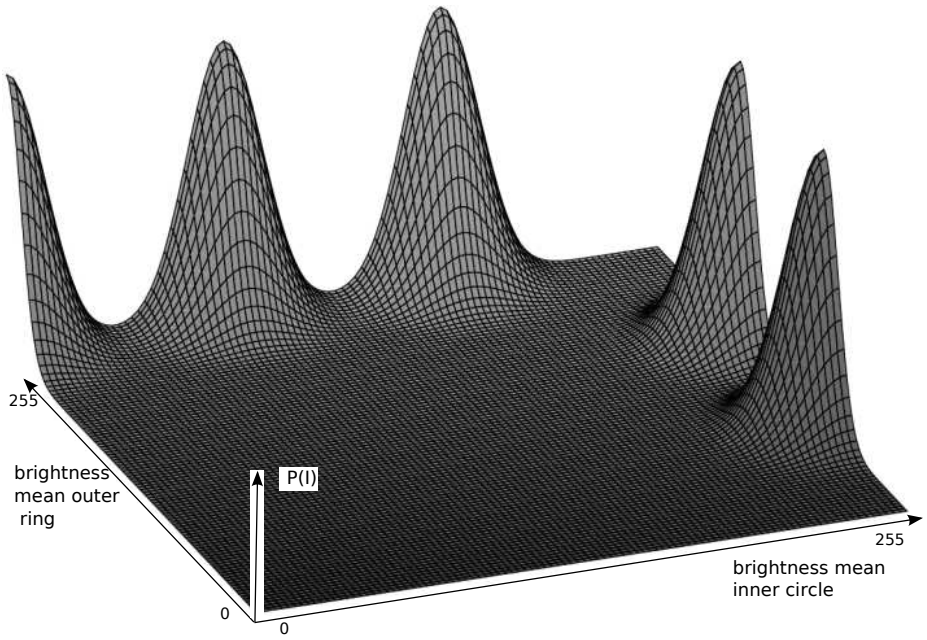


FIGURE 5.7: Identification prior for relative pixel brightness

MAP decoding using Bayesian estimation Next is to choosing a decision making algorithm to decode the blobs. Each blob represents one letter of an alphabet: one out of $a = 5$ hypothesis is true, the others are false. We add a sixth hypothesis, that the feature is too damaged to decode. We calculate the probability of each of the hypothesis h given the observations $\mathbf{z}$: $P(H = h|\mathbf{Z} = \mathbf{z})$, abbreviated $P(h|\mathbf{z})$. This discrete PDF is then reduced to the most probable value using a MAP (Maximum A Posteriori) strategy. Thus, the hypothesis that has the maximum probability decides on the code transmitted to the next processing step.

To calculate these probabilities, we use Bayes' rule:

$$P(h|\mathbf{z}) = \frac{P(\mathbf{z}|h)P(h)}{P(\mathbf{z})} \tag{5.2}$$

Since we are only interested in the maximum of this PDF and not its absolute values, we can omit the denominator: it is independent of the hypothesis, and thus only a scaling factor.
Let n_h be the number of features of type h in the pattern, then for $h = 0..4$:

$$P(H = h) = P(h) = \frac{n_h}{\sum_{i=0}^{4} n_i}$$

$P(H = 5)$ is the probability to be dealing with a blob that passed all previous consistency tests but still cannot be decoded. This probability needs to be estimated empirically, e.g. choose $P(H = 5) = 0.005$ for a start. One factor of equation 5.2 remains to be calculated: $P(\mathbf{z}|h)$:

$$P(\mathbf{z}|h) = P(\mathbf{z}|\mathbf{i})P(\mathbf{i}|h) \Rightarrow P(h|\mathbf{z}) \sim P(\mathbf{z}|\mathbf{i})P(\mathbf{i}|h)P(h)$$

The distribution of $P(h)$ is already clear, the two other factors can be calculated as follows:

- $P(\mathbf{i}|h)$: given a PDF with only one of the Gaussians of figure 5.7, how probable is the scaled ratio of the means of the pixel values in the blob. Since the Gaussians in figure 5.7 have been chosen sufficiently apart, $P(\mathbf{i}|h) \approx P(I)$: one can use the distribution of equation 5.1.

- $P(\mathbf{z}|\mathbf{i})$: given the segmentation between inner circle and outer ring, what is the probability that the pixel values belong to that segmentation.
  The EM-algorithm segments both parts of the feature. Thus given is the resulting Gaussian PDF (mean and standard deviation) of inner circle and outer ring. For efficiency reasons, the pixel values in the blob are represented by the median of the inner circle pixels and the median of the outer ring pixels. Thus $P(\mathbf{z}|\mathbf{i})$ requires only two evaluations of a normal distribution function.
  $P(\mathbf{z}|\mathbf{i}) = \mathcal{N}(\mu_{in}, \sigma_{in}) + \mathcal{N}(\mu_{out}, \sigma_{out})$

5.2.3 Feature tracking

For efficiency reasons, this feature initialisation procedure is not performed every frame. To avoid this one can use a Bayesian filter to track the blobs. The system can handle occlusions: when an object is occluded by another, feature tracking is interrupted, and the feature is re-initialised.

Situations with considerable clutter, where similar features are close to one another, require a **particle filter**, also known as sequential Monte Carlo methods. The corresponding PDF allows to account for multiple hypothesis, providing more robustness against clutter and occlusions. Every filter is based on one or more low level vision features. Isard and Blake [1998] proposes to use edges as low level feature input. He shows for example that in this way his *condensation algorithm* (a particle filter) can track leaves in a tree. Nummiaro et al. [2002] on the other hand, uses colour blobs instead of edges as an input for her particle filter. The application determines the low level vision feature required: environments can either have strong edges, or strong colour differences etc. Nummiaro et al. [2002] shows that her technique performs well for e.g. people tracking in crowds, also a situation where the computational overhead of a particle filter pays off as there is a lot of clutter.

For this structured light application, visual features are well separated, and the extra computational load of particle filtering would be overkill: this segmentation does not need the capability to simultaneously keep track of multiple hypotheses. A **Kalman filter** is the evident alternative approach to track blobs over the video frames. However, a KF needs a motion model, for example a constant velocity or acceleration model. Since the 2D motion in the image is a perspective projection of motion in the 3D world, using a constant velocity or acceleration model in the 2D camera space is often not adequate. The interesting (unpredictable) movements and model errors are modelled as noise and the KF often fails.

Therefore, this thesis employs the **CAMShift** object tracking algorithm, as proposed by Bradski [1998]. This is a relatively robust, computationally efficient tracker that uses the mean shift algorithm. This relatively easy tracking situation does not require a more complex tracking method. First it creates a histogram of the blob of interest. Using that histogram the algorithm then calculates the backprojection image. For each pixel backprojection replaces the value of the original image with the value of the histogram bin. Thus the value of each pixel is the probability of the original pixel value given the intensity distribution. Thresholding that backprojection image relocates the blob in the new image frame: use a local search window around the previous position of the blob.

Authors like Bouguet [1999] and François [2004] propose another interesting procedure to track blobs: using a multi-resolution approach. Selecting the appropriate subsampled image at each processing step drastically reduces the computing time needed.

5.2.4 Failure modes

We now discuss under which conditions the proposed system fails, and whether those conditions can be detected online :

- Condition: When the **scene** is **too far** away from the end effector **in comparison to the baseline**. Then triangulation is badly conditioned. Section 5.4.2 discusses the sensitive combinations of calibration parameters in detail. The angle θ is the angle between camera and projector, and thus represents the distance to the scene relative to the baseline. Based on the required accuracy, section 5.4.2 provides equations to determine if this desired accuracy is achieved for a certain θ, in combination with other system parameters. Clearly values of θ near 0 or π are unacceptable.
  Detection: Since the setup is geometrically calibrated, one known at any point in time online what the position of projector and camera is. The technique of *constrained based task specification* (see section 6.2) allows to add a constraint that avoids that the principle ray of camera and projector become near parallel. This constraint can be added such that it is binding, or such that it is advisory (depending on the relative constraint weighting).

- Condition: When the scene is **self occluded**: when the end effector moves in the light pyramid of the projector, the projected features are occluded by the robot itself. The projector needs to be positioned such that this situation does not occur for the application at hand.
  Detection: as in the previous case.

- Condition: When the **scene** is **too far away**. Without an external physical length measurement, the entire 3D reconstruction is only known up to a scale factor, see section 4.4.1. Hence, image enlarging the setup, then nothing changes in the camera image except for the intensity of the blobs. Therefore, the distance between projector and scene also has physical limitations. The illuminated area is equal to $\frac{z_p^2 \mathcal{W}_p \mathcal{H}_p}{f_p^2}$. Assuming a 1500 $lumen$ projector and a limit to the blob illumination of 500 lux, the maximal distance $z_p = \sqrt{\frac{1500 lumen \; 2000^2 pix^2}{500 \frac{lumen}{m^2} \; 768 pix \; 1024 pix}} \approx 4m$.
  Detection: as in the previous case.

- Condition: When the **assumption** that the scene is **locally planar** is **violated**. The system assumes that for most of the blobs, the surface they illuminate is near planar. Then the circle deforms as an ellipse, and segmentation will succeed. However, some surfaces are very uneven, like for example a heating radiator: in those cases this structured light system will fail. Time-multiplexed structured light may be a solution in such cases, as each projection feature needs a much smaller surface there (see section 3.3.4).
  Consider the projector pinhole model on the left of figure 5.8. A blob in

the projector image produces a cone of light. This cone is intersected with a locally planar part of the scene at distance z_p. The area of this intersection depends on the relative orientation between scene and projector, characterises by angle α. It also depends on the location of the blob in the projector image, at an angle β with the ray through the principal point. As these two reinforce each other in increasing in the ellipse surface, consider the case for a blob in the centre of the projection image, to simplify the equations. Then the cone is determined by $x^2 + y^2 = \frac{D_u^2}{4f^2} z^2$. The plane under angle α by $z = \tan(\alpha)y + z_p$. Combining these equations results in the conic section $x^2 + (1 - e^2)y^2 - 2py + p^2 = 0$ where the eccentricity $e = \frac{D_u \tan\alpha}{4f^2}$, and the focus of the section is at $(0, p)$. For $0 < e < 1$ the intersection is an ellipse. Indeed, since f is of order of magnitude 10^3 and $D_u \approx 10$, $e > 1$ for $\alpha > 1.566$: larger than $89.7°$, which is an impossible situation. The equation can be rewritten in the form:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1 \text{ with } (x_0, y_0) = (0, \frac{z_p D_u^2 \tan\alpha}{4f_p^2 - D_u^2 \tan^2\alpha}) \text{ and}$$

$$(a, b) = (\frac{D_u z_p \sqrt{4f_p^2 + 2D_u^2 \tan^2\alpha}}{2f_p \sqrt{4f_p^2 - D_u^2 \tan^2\alpha}}, \frac{D_u z_p \sqrt{4f_p^2 + 2D_u^2 \tan^2\alpha}}{4f_p^2 - D_u^2 \tan^2\alpha})$$

This is the equation of the projection of the ellipse in the xy plane, therefore its area is

$$A = \pi ab \cos\alpha = \frac{\pi D_u^2 z_p^2 (4f^2 + 2D_u^2 \tan^2\alpha)}{2f \cos\alpha (4f^2 - D_u^2 \tan^2\alpha)^{\frac{3}{2}}}$$

For example, take $f_p = 2000pix$ and $z_p = 1m$, see section 5.4.2.3. For the top right pattern of figure 3.7, the number of columns $c = 48$. Let d be the unilluminated space between the blobs, then:

$$\mathcal{W}_p = c(d + D_u) \overset{d=D_u}{\Rightarrow} D_u = \frac{1024pix}{2 \cdot 48} \approx 11pix$$

If one assumes the space between two blobs to be equal to the diameter of the blobs. The right hand side of figure 5.8 plots the area in m^2 in function of D_u and α.

Also the industrial and surgical application discussed of the **experiments** chapter, the scene is not always locally planar. As a solution these experiments propose to combine this type of structured light with a different, model-adapted type:

- The burr removal application of section 8.3: the surface of revolution is locally planar, except for the burr itself: use two phases of structured light: the proposed sparse 2D pattern, and a model adapted 1D pattern.

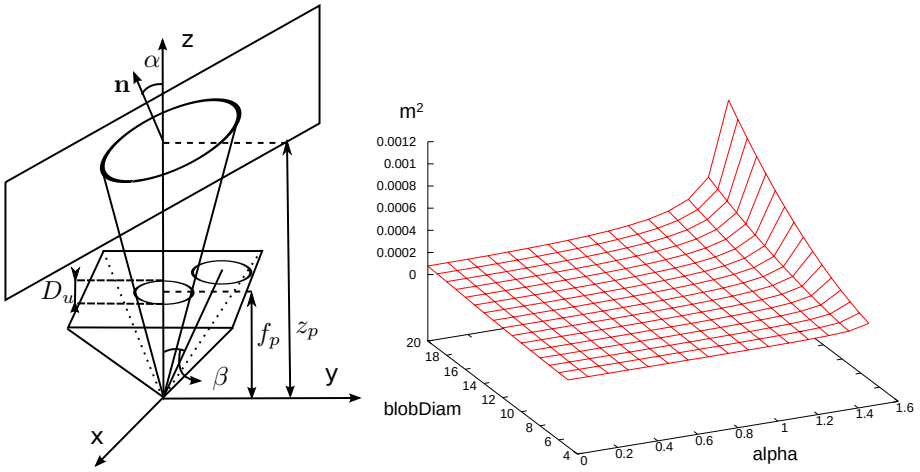


FIGURE 5.8: Left: projector pinhole model with blob in centre of image, right: area of the ellipse on the scene

– The automation of the surgical tool in section 8.4: also here the surface is locally planar, except for the wound: the two phases of structured light proposed are similar: first the sparse 2D pattern, then a (different) model adapted 1D pattern.

Thus, for these cases, adapting the size of the blobs is insufficient, one has to **adapt the type of pattern** to achieve the required resolution.
Detection: It is hard to determine whether a blob is deformed because of a local discontinuity, or because of other reasons. But the deformed blob do not contain the necessary information any more anyway, do determining the reason for the deformation is futile.

- Condition: When the **assumption** that the **reflectivity** is **locally constant** is violated. If the scene is locally highly textured within a projected blob, segmentation may fail. Or in other words: a wide variety of different reflectivities within one of the illuminated areas can make the decoding process fail.
  Detection: Identical to the previous case.

- Condition: When **external light sources look like the projected patterns**. This external light source then has to comply to the following demands:

  – Be powerful. Normal ambient light is negligible in comparison to the projected light. Indeed, the projector produces ≈ 1500 *lumen*, if it lights up an area of $1m^2$, the illuminance is 1500 *lux* (see section 3.3.3). The LCD attenuates part of the projected feature, but the part that produces a near maximal output in the camera image does

reach a illumination this strong (see section 3.3.6). Compare this to the order of magnitude 50 *lux* for a living room, 400 *lux* for a brightly lit office, see section 5.2.1. Indirect sunlight for example may happen to be in the same illumination range.

- Be ellipse shaped and of similar size. This is thinkable, indirect sunlight that passes trough an object with circular openings that happen to be just about large enough.
- Consist of two intensities in the following way. Consider an ellipse with the same centre, and a semimajor and semiminor axis with a length of about 70% of the original lengths. Then the area within this ellipse needs to be about uniform in intensity, and the area outside of it also, but with a different intensity.

The probability that this happens in combination with the previous demands seems very small. Therefore this is not a realistic failure mode. Detection: Identical to the previous case.

5.2.5 Conclusion

We propose a method to segment ellipsoid blobs in the camera image. The strong point of this method is its functioning in different environments, its robustness: against different illumination, against false positives and partly against depth discontinuities. The reason for this robustness is that the segmentation pipeline is threshold free: all thresholds are data-driven. This does not mean that all blobs will always be segmented correctly (see section 5.2.4): the appropriate information needs to be present in the camera image.
The disadvantages are the larger processing time (e.g. P-tile vs EM segmentation) and the rather low resolution. Section 7.3, named *achieving computational deadlines through hard-and software*, discusses solutions when the former would pose a problem. The latter is not a problem in robotics applications, as iterative refinement during motion provides sufficient accuracy.

5.3 Labelling

5.3.1 Introduction

Whereas segmentation (section 5.2) decodes a single blob, labelling decodes the combinations of those blobs that form codewords: the submatrices. It takes the result of the segmentation process as input, and performs data association between neighbouring blobs to find the w^2-tuples of blobs. (in the experiments we use $w = 3$) The result is the correspondences between camera and projector. The algorithm is also parameter free: all needed parameters are estimated automatically, no manual tuning is needed.

5.3.2 Finding the correspondences

Algorithm 5.1 shows the labelling procedure. We explain each of these steps here:

Algorithm 5.1 label(B) with B = segmented blobs

```
preProcess(){Executed only once}
if features cannot be tracked in this frame then
  find4Closest(B)
  addDiagonalElems(B)
  testConsistency(B)
  findCorrespondences(B)
  undistort(B)
end if
```

- *preProcess*: the result of the labelling procedure is a string of w^2 base a values for every segmented blob. Each of those strings must be converted to a 2D image space position in the projector image. We generate a LUT to accelerate this process: it is better to spend more computing time in an offline step than during online processing.

  For every projected blob, generate its string of w^2 base a values out of the blob itself and its $w^2 - 1$ neighbours. These strings are only defined up to a cyclic permutation. Hence, generate all cyclic permutations as well. Then sort this list on increasing string value, keeping their associated u and v projector coordinates. (introsort was used for this sorting, but efficiency is not important in this step, since it is done offline)

- *find4Closest*: We now detect the four nearest neighbours: the ones that are left, right, above and below the central blob in the projector image. To that end, for each blob, take the subset of segmented blobs of which the u and v coordinates are both within a safe upper limit for the neighbours of the blob (this step was added for efficiency reasons). Within that subset,

find the 4 closest blobs such that they are all $\approx \frac{\pi}{2}$ apart: take $> \frac{\pi}{3}$ as a minimum. For algorithm details, see appendix B.1.

- *addDiagonalElems*: For each blob i, based on the 4 neighbours found in the previous step, predict the positions of the diagonal elements in the camera image (a factor $\sqrt{2}$ further along the diagonals). Then correct the predicted positions to the closest detected blob centres in the camera image. Name the 8 neighbours $N_{i,j}$ for $0 \leq j \leq 7$.

- *testGraphConsistency*: Test whether the bidirectional graph that interconnects the detected features is consistent: do adjacent features of a feature point back at it. Thus, for each blob i, and for each of its neighbours j, find the neighbour k that points back at it: of which the angle made by the vector from the centre of j to the neighbour k is at $\approx \pi$ from the angle made by the vector from the centre of i to neighbour j. One can perform the following consistency checks:

  - Do all 8 neighbours j also have blob i as a neighbour? (8 restrictions)
  - Are all 8 neighbours j correctly interconnected among themselves? Or in other words, suppose we would turn the submatrix such that one of its closest neighbours is along a vertical line, is the north-east neighbour then equal to the east neighbour of the north neighbour, and equal to the north neighbour of the east neighbour? (16 restrictions)

  If one of these 24 constraints is not satisfied, then reject the corresponding blobs. See algorithm B.2 for more details.

- *findCorrespondences*: For each of the approved blobs, look up the corresponding (u,v) projector coordinate in the LUT generated in the preprocessing step. Since this LUT is sorted, and all possible rotations are accounted for, a binary search algorithm can do this in $O(\log n)$ time.

  This step also enforces decoding consistency using a **voting** strategy. Morano et al. [1998] describe the voting used here: since each blob is part of w^2 (9 in the experiments) submatrices, it receives up to w^2 votes if all submatrices are valid, less if some are not. Assuming $h = 3$, the confidence number is higher, since each of these submatrices can be labelled w^2 times, each time disregarding one of its elements. This leads to a total maximum confidence number of w^4 (in this case 81). The final decoding is determined by the code that received the maximum number of votes, see algorithm B.3 for details.

- *undistort*: We compensate for radial distortions. We do this after labelling, for efficiency reasons. Since we are only interested in the $r \times c$ projected points, we only undistort these points and not all $\mathcal{W}_c\mathcal{H}_c$ pixels. Section 4.3.3 explains what distortions are accounted for and how.

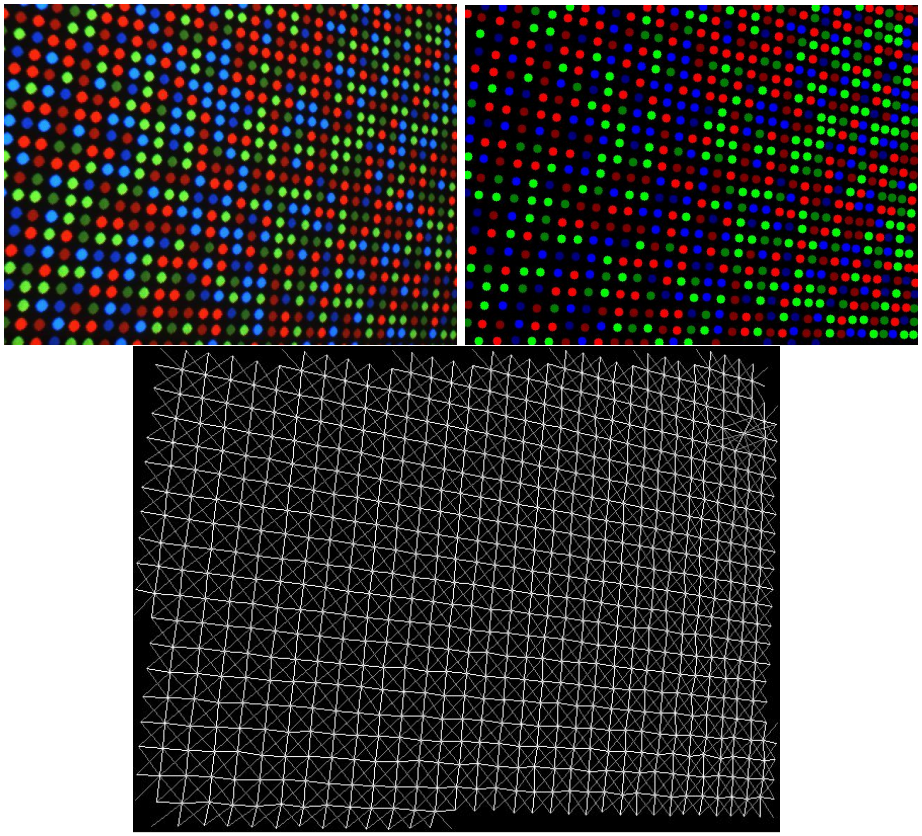


Figure 5.9: Labelling experiment. Top left: the image, top right: the segmentation, bottom: the labelling

Figure 5.9 is an example of the result of this labelling procedure. It uses the technique of section 8.2 on a near flat surface: on the top left the captured camera image, and the top right part shows a synthetic image containing the blob segmentation. This experiment uses coloured features, but the concentric circle features of section 3.3.6 could have been used here equally fine: the only difference is in the segmentation, not in the labelling.
The bottom of figure 5.9 shows the labelling: observe that in the upper right corner, where the blobs are closest together, the 8 closest neighbours of several of the elements are inconsistent. The graph consistency check detects this problem, and rejects the inconsistent blobs.

5.3.3 Conclusion

This section described a procedure for decoding the structure among the segmented blobs. The system is relatively robust thanks to the use of model knowledge:

- The physical structure of the matrix: each of the 4 diagonals are $\sqrt{2}$ further away than the 4 closest neighbours, and in a certain predicted direction. The regularity of the grid (neighbouring arrows need to point in both directions) can be checked.
- The properties of the matrix: through a voting algorithm, one error can be corrected in every submatrix.

5.4 Reconstruction

This section uses the calibration parameters estimated in section 4 to reconstruct a point cloud online. Paragraph 5.4.1 formulates the reconstruction algorithm, while paragraph 5.4.2 evaluates the accuracy of the 3D sensor.

5.4.1 Reconstruction algorithm

The equations

The 6D orientation of the camera is determined by a translation vector $\mathbf{t}_c^w$ and a rotation matrix $\mathbf{R}_c^w$ that describe the position and orientation of the camera expressed in the world frame, similarly for the projector. Let (x_j, y_j, z_j) be the coordinates of the reconstructed point j in the world frame, then equation 4.3 holds, for $i = p, c$:

$$\rho_{i,j} \begin{bmatrix} u_{i,j} \\ v_{i,j} \\ 1 \end{bmatrix} = \mathbf{P}_i \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \tag{5.3}$$

Equation 4.3 represents a ray through the *hole* of the camera (projector) pinhole model and the point $\mathbf{u_i}$ in the camera (projector) image: the first two equations are equations for (perpendicular) planes that intersect in this ray. The third equation of each system of equations eliminates the unknown scale factor $\rho_{i,j}$. What remains are 4 equations for 3 unknowns (the 3D Cartesian coordinates): in theory, these rays intersect in space, and then there is a linear dependence in the equations. In practice the rays cross, and this overdetermined system of equations is inconsistent.

Let $\mathbf{P}_i = \begin{bmatrix} \mathbf{P}_{i,1} \\ \mathbf{P}_{i,2} \\ \mathbf{P}_{i,3} \end{bmatrix}$ for $i = c, p$. Equation 5.3 $\Rightarrow \rho_{i,j} = \mathbf{P}_{i,3} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \Rightarrow$

$$\begin{cases} \mathbf{P}_{i,3} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} u_i = \mathbf{P}_{i,1} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \\ \mathbf{P}_{i,3} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} v_i = \mathbf{P}_{i,2} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \end{cases} \Rightarrow \begin{bmatrix} u_c\mathbf{P}_{c,3} - \mathbf{P}_{c,1} \\ v_c\mathbf{P}_{c,3} - \mathbf{P}_{c,2} \\ u_p\mathbf{P}_{p,3} - \mathbf{P}_{p,1} \\ v_p\mathbf{P}_{p,3} - \mathbf{P}_{p,2} \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \equiv \mathbf{A} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} = 0 \tag{5.4}$$

Geometrically, by solving this system of equations, we find the centre of the shortest possible line segment between the two rays, indicated by the (red) cross

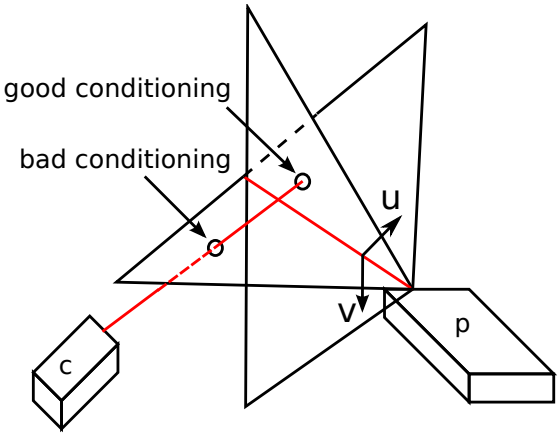


Figure 5.10: Conditioning of intersection with (virtual) projector planes corresponding to projector ray

in figure 4.14. As straightforward methods are computationally expensive to calculate online, one could think of simply dropping one of the equations, removing the overdetermination of the system of equations. Geometrically this corresponds to a ray-plane triangulation instead of a ray-ray triangulation. However, some of the equations may have a good conditioning number, and other a very bad one. See figure 5.10: suppose the ray is in the camera pinhole model, and the plane in the projector pinhole model (the other way around is equivalent). If the u-axis in the image of camera and projector are in a horizontal plane (or close to it), the projector plane that is well conditioned is the one determined by the u-coordinate of the projector: third equation in the system of equations 5.4. The plane determined by the v-coordinate (fourth equation there) is badly conditioned. This is mathematically the same situation as in figure 3.5. As the robot end effector, and thus the camera, can have any 6D orientation relative to the projector, which equations are well conditioned and which are not changes during the motion of the robot. Therefore we cannot simply omit any of these equations.

Equation 5.4 is a non-homogeneous system of equations, as the last element of $\begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix}$ is not unknown. Hence, it can be transformed to:

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}_{1:3} & \mathbf{A}_4 \end{array} \right] \Rightarrow \mathbf{A}_{1:3}\mathbf{x} = -\mathbf{A}_4 \Rightarrow \mathbf{x} = -\mathbf{A}_{1:3}^{\dagger}\mathbf{A}_4$$

A solution using SVD

As the rows of $\mathbf{A}$ are linearly independent, an explicit formula for the Moore-Penrose pseudo-inverse exists, but calculating it is an expensive operation. It is not very suitable if we have to compute this for every point in every image frame. Therefore we choose a different approach to solve this least squares problem:

return to the system of equations in homogeneous space (equation 5.4) but add an extra variable η as the fourth coordinate of the unknown vector. The problem to be solved is then a special case of these equations for $\eta = 1$: $\mathbf{A}\begin{bmatrix}\mathbf{x}\\ \eta\end{bmatrix} = 0$. Let $\mathbf{x}_h \equiv \begin{bmatrix}\mathbf{x}\\ \eta\end{bmatrix}$. In order to avoid the trivial solution $x = y = z = \eta = 0$, we add an extra constraint: $\|\mathbf{x}_h\| = 1$. Then:

$$\begin{aligned} &\text{minimise } \|\mathbf{A}\mathbf{x}_h\| \text{ under } \|\mathbf{x}_h\| = 1 \overset{\text{SVD decomposition}}{\Rightarrow} \\ &\arg\min_{\mathbf{x},\eta} \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{x}_h\| \text{ with } \|\mathbf{V}^T\mathbf{x}_h\| = 1 \end{aligned}$$

where $\mathbf{V}$ contains a set of orthonormal vectors. The vectors in $\mathbf{U}$ are also orthonormal, therefore $\arg\min_{\mathbf{x},\eta} \|\Sigma\mathbf{V}^T\mathbf{x}_h\|$ yields the same solution. Substitute $\mathbf{\Upsilon} = \mathbf{V}^T\mathbf{x}_h$, then the problem is equivalent to

$$\arg\min_{\mathbf{x},\eta} \|\mathbf{\Sigma}\mathbf{\Upsilon}\| \text{ with } \|\mathbf{\Upsilon}\| = 1$$

As $\mathbf{\Sigma}$ is diagonal and contains singular values in descending order, the solution is $\mathbf{\Upsilon}^T = [0\ \ldots\ 0\ 1]$. Since $\mathbf{x}_h = \mathbf{V}\mathbf{\Upsilon}$, the last column of $\mathbf{V}$ is the correct solution. Thus, the right singular vector corresponding to the smallest singular value of $\mathbf{A}$, $\mathbf{V}_4$, is proportional to $\begin{bmatrix}\mathbf{x}\\ \eta\end{bmatrix}$. Divide the vector by η to convert the homogeneous coordinate to a Cartesian one.

A solution using eigenvectors

Since we do not need the entire SVD, but only $\mathbf{V}_4$, calculating a singular value decomposition for every point in every image frame is unnecessarily expensive. Therefore convert the problem to an eigenvalue problem :

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$$

The smallest eigenvalue and corresponding eigenvector of $\mathbf{A}^T\mathbf{A}$ are respectively equal to the smallest singular value of $\mathbf{A}$, and its corresponding right singular vector. As $\mathbf{A}$ is 4×4, $|\mathbf{A} - \lambda\mathbf{I}|$ is a 4th degree polynomial, of which the roots can be calculated analytically: $|\mathbf{A} - \lambda\mathbf{I}| = 0$ is a quartic equation with an analytical solution using Ferrari's method. For unchanged relative positioning between robot and camera, the projection matrices remain unchanged, and thus this solution can be parametrised in the only variables changing: $\mathbf{u_c}$, $\mathbf{u_p}$, to accelerate processing. We find $\mathbf{V}_4$ by solving $(\mathbf{A} - \lambda_4\mathbf{I})\mathbf{V}_4 = 0$: subtracting the eigenvalue converted the regular matrix into a singular one. Since any row thus is linearly dependent on a combination of the other 3 rows, we can omit one. Let $\mathbf{B} = \mathbf{A} - \lambda_4\mathbf{I}$, $\eta = 1$ then:

$$\mathbf{B} = \left[\begin{array}{c|c} \mathbf{B}_1^{(3\times3)} & \mathbf{B}_2^{(3\times1)} \\ \hline \mathbf{B}_3^{(1\times3)} & B_4 \end{array}\right] \Rightarrow \mathbf{B}_1\mathbf{x} = -\mathbf{B}_2$$

The last equation can be solved by relatively cheap Gauss-Jordan elimination: $O(n^3)$ for a square $n \times n$ $\mathbf{B}_1$, but with n as small as 3. This saves considerably in processing load compared to the iterative estimation of a SVD decomposition.

5.4.2 Accuracy

The aim of this section is to estimate the robot **positioning errors** as a function of the measurements and the different system parameters. Usually robot tasks are speficied with a required accuracy. Using this sensitivity analysis, this required mechanical accuracy can be **translated into an accuracy on measurements or parameters**. The structured light setup can be adapted accordingly.

5.4.2.1 Assumptions

Chang and Chatterjee [1992] (and later Koninckx [2005]) perform an error analysis of a structured light system assuming that the lines from the focal point to the centre of the image planes in camera and projector are coplanar. Then the geometry of the setup can be projected in a plane, and hence becomes 2D instead of 3D, see figure 5.11. This assumption can easily be approximately satisfied in a setup with a fixed baseline as both imaging devices are usually mounted on the same planar surface. These error analysis neglect many of the contributions caused by calibration errors, although they can be substantial. Therefore we do incorporate these in this analysis.

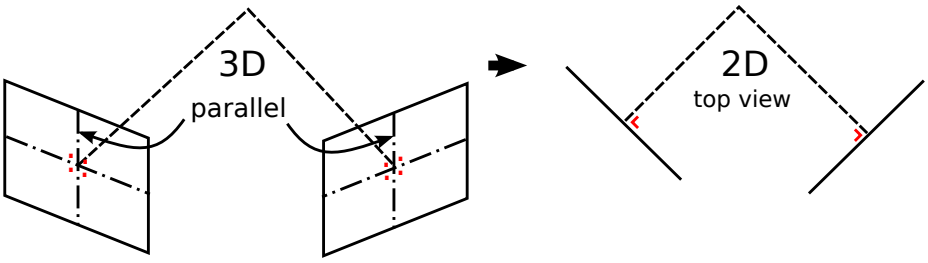


Figure 5.11: Coplanarity assumption of accuracy calculation by Chang

In the structured light application studied here, the assumption depicted in figure 5.11 is not valid: the transformation from the frame attached to the camera to the frame attached to the projector is an arbitrary 6D transformation. We analyse the error for this case. To simplify the derivations assume a simple pinhole model: the principal point in the centre of the image, the image axes perpendicular and the principal distances in equal in both directions. This is an assumption that is reasonably close to reality for accuracy calculations.

5.4.2.2 Accuracy equations

Let the coordinate of a point to be reconstructed be $\{x_c, y_c, z_c\}$ in the camera frame, and $\{x_p, y_p, z_p\}$ in the projector frame. $\{x_b, y_b, z_b\}$ is the vector defining the baseline, then for $i = p, c$:

$$x_i = \frac{u_i z_i}{f_i} \qquad y_i = \frac{v_i z_i}{f_i} \qquad \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{R}(\psi,\theta,\phi) & \begin{matrix} x_b \\ y_b \\ z_b \end{matrix} \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

with the origin of the image coordinates in the centre of the image, R the 3DOF rotation matrix (expressed in Euler angles ψ, θ and ϕ for example). To simplify the notation here, all image coordinates (u, v) are corrected for radial distortion, avoiding the subscript u.

$$\begin{cases} R_{11}x_c + R_{12}y_c + R_{13}z_c + x_b = \dfrac{u_p(R_{31}x_c + R_{32}y_c + R_{33}z_c + z_b)}{f_p} \\ x_c = \dfrac{u_c z_c}{f_c} \\ R_{21}x_c + R_{22}y_c + R_{23}z_c + y_b = \dfrac{v_p(R_{31}x_c + R_{32}y_c + R_{33}z_c + z_b)}{f_p} \\ y_c = \dfrac{v_c z_c}{f_c} \end{cases} \tag{5.5}$$

Substituting the second and last equation in the first equation yields:

$$z_c = \frac{f_c(z_b u_p - x_b f_p)}{u_c f_p R_{11} + v_c f_p R_{12} + f_p f_c R_{13} - u_p u_c R_{31} - u_p v_c R_{32} - u_p f_c R_{33}} \tag{5.6}$$

z_c is dependent on the image coordinates of the point u_c, v_c, u_p, the intrinsic parameters f_c, f_p and the extrinsic parameters $x_b, z_b, \psi, \theta, \phi$. Of these variables only u_p is known exactly, all others are (imperfectly) estimated.
Hence, the uncertainty on z_c is $\Delta Z_c \approx \sqrt{E_{coord} + E_{intr} + E_{extr}}$ where

$$E_{coord} = (\frac{\partial z_c}{\partial u_c}\Delta u_c)^2 + (\frac{\partial z_c}{\partial v_c}\Delta v_c)^2$$

$$E_{intr} = (\frac{\partial z_c}{\partial f_c}\Delta f_c)^2 + (\frac{\partial z_c}{\partial f_p}\Delta f_p)^2$$

$$E_{extr} = (\frac{\partial z_c}{\partial x_b}\Delta x_b)^2 + (\frac{\partial z_c}{\partial z_b}\Delta z_b)^2 + (\frac{\partial z_c}{\partial \psi}\Delta \psi)^2 + (\frac{\partial z_c}{\partial \theta}\Delta \theta)^2 + (\frac{\partial z_c}{\partial \phi}\Delta \phi)^2$$

This high dimensional error function can be used in the paradigm of constraint based task specification [De Schutter et al., 2005]. The larger the error in function of the current robot pose, the less one would like the robot to be in that position. Use this quality number in the weights of a constraint of the task specification to avoid the poses where the structured light is least robust. Although this way one only stimulates a local improvement in structured light

conditioning, that may correspond to a local minimum of the error function. But then again, constraint based task specification also only specifies a instantaneous motion, and has no path planning. Therefore, it is advisable not to overestimate the priority of this constraint, in order not to get stuck in local minima of the error function. Moreover, the robot task is more important than the behaviour of the sensor: when one sensor is not useful in a certain configuration, others can take over.

5.4.2.3 The contribution of pixel errors

First, calculate the contribution of the error in the pixel coordinates:

$$\frac{\partial z_c}{\partial u_c} = \frac{z_c^2(u_p R_{31} - f_p R_{11})}{f_c(z_b u_p - x_b f_p)}$$

$$\frac{\partial z_c}{\partial v_c} = \frac{z_c^2(u_p R_{32} - f_p R_{12})}{f_c(z_b u_p - x_b f_p)}$$

For example, for a pixel in the centre of the projector image ($u_p = 0$)

$$E_{coord} = (\frac{z_c^2}{f_c x_b})^2[(R_{11}\Delta u_c)^2 + (R_{12}\Delta v_c)^2]$$

with $R_{11} = \cos(\psi)\cos(\phi) - \cos(\theta)\sin(\phi)\sin(\psi)$, $R_{12} = -\cos(\psi)\sin(\phi) - \cos(\theta)\cos(\phi)\sin(\psi)$ ($z - x - z$ convention for Euler angles). $|\Delta u_c|$ and $|\Delta v_c|$ are due to erroneous localisation of the projected blobs. Worst case scenario, the localisation is off in both directions, hence take for example $|\Delta u_c| = |\Delta v_c|$:

$$E_{coord} = (\frac{z_c^2 \Delta u_c}{f_c x_b})^2[1 - sin^2(\psi)sin^2(\theta)] \tag{5.7}$$

It increases dramatically (z_c squared) for increasing depth: the further away the objects, the more prone to errors the reconstruction is. But as z_c is a function of the other variables, substitute z_c to make E_{coord} only dependent on the base variables.

$$E_{coord} = (\frac{f_c x_b \Delta u_c}{(u_c R_{11} + v_c R_{12} + f_c R_{13})^2})^2[1 - sin^2(\psi)sin^2(\theta)] \tag{5.8}$$

E_{coord} logically increases when the pixel offset increases. It also logically increases with an increasing baseline: increasing the scale of the setup increases all physical distances, also the errors.

Equation 5.8 is a function of all 3 Euler angles. A 4D cut of this 9D function cannot to be visualised, but a 3D cut can by fixing one of the parameters. For the principal point of the camera for example, the function is only dependent on 2 Euler angles: θ and ψ:

$$E_{coord}(u_c = 0, v_c = 0) = (\frac{\Delta u_c}{f_c})^2(\frac{1}{[\sin(\psi)]^2[\sin(\theta)]^2} - 1)$$

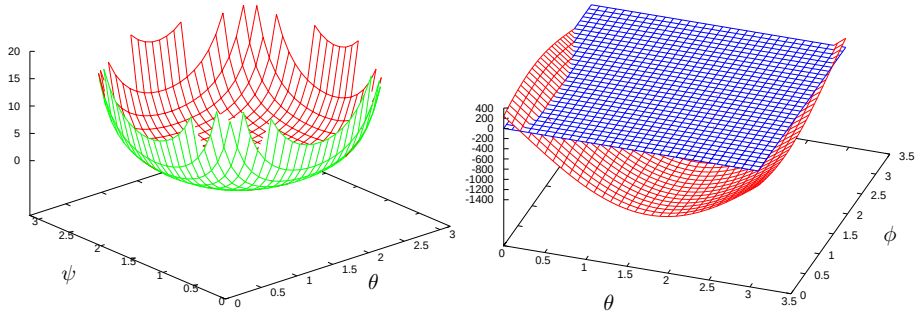


FIGURE 5.12: Left: error scaling for $u_c = v_c = 0$ pix, right: denominator of partial derivative for $\psi = \frac{\pi}{2}$

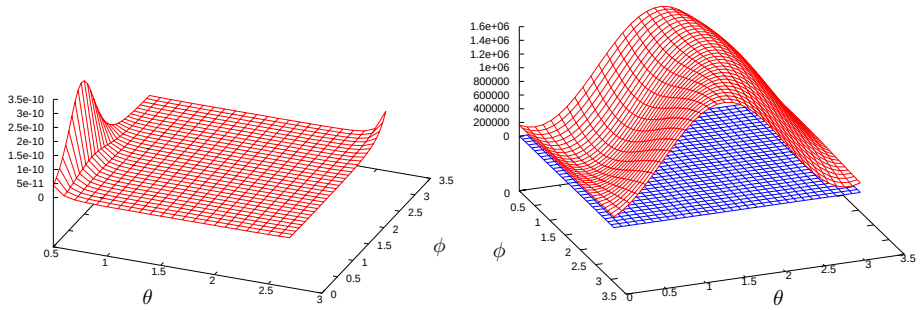


FIGURE 5.13: Left: second factor of partial derivative for $\psi = \frac{\pi}{2}$, right: function $\mathcal{E}$, sum of squared denominators

The left graph of figure 5.12 shows the error scaling due to the orientation in this case. For $\psi = i\pi$ for $i \in \mathbb{Z}$, $E_{coord}(u_c = 0, v_c = 0)$ becomes infinitely large: the sine in the denominator the last equation is 0. This is because ψ is the angle that determines the conditioning of the ray-plane intersection (see figure 5.10). For $\psi \approx 0$ the ray and the plane are about parallel, hence its conditioning is bad. The best conditioning is reached for $\psi = \frac{\pi}{2} + i\pi$ for $i \in \mathbb{Z}$: then the camera ray and projector plane are perpendicular. (although in practice, this is an unimportant effect as it will be attenuated by solving the system of equations with least squares).
Also for $\theta = i\pi$ for $i \in \mathbb{Z}$ $E_{coord}(u_c = 0, v_c = 0)$ becomes infinite: parallel z-axes of the imaging devices make triangulation impossible.

Now consider any point of the image under the favourable circumstance that $\psi = \frac{\pi}{2}$. This a reasonable assumption, since this error calculation is only based on 3 of the 4 equations in the system of equations 5.5: if the ray-plane triangulation defined by those 3 equations is badly conditioned because of ψ, the fourth plane equation will be well conditioned and the error will be modest through the least squares solution. Then:

$$E_{coord}(\psi = \frac{\pi}{2}) = (f_c x_b \Delta u_c)^2 \frac{1 - [\sin(\theta)]^2}{(u_c \cos(\theta)\sin(\phi) + v_c \cos(\theta)\cos(\phi) - f_c \sin(\theta))^4} \tag{5.9}$$

In this case the error is mainly determined by the possibility of the denominator becoming 0. The right hand side of figure 5.12 plots the quartic root of the denominator of equation 5.9 for $u_c = (u_c)max$, $v_c = (v_c)max$, and the plane $z = 0$. The intersection of these surfaces determines a 2D curve of angle combinations that make the denominator 0, and hence make the error infinite. Fortunately, all these combinations fall in the range where the error contribution due to pixel deviations is large, and hence pose no extra restrictions: they are near $\theta = 0$ (maximum 17°) or $\theta = \frac{\pi}{2}$, with θ the angle between the z-axes of the imaging devices.
Therefore the left hand side of figure 5.13 presents a 3D cut of the second factor of equation 5.9, where values of θ near 0 and π have been omitted.

Logically, ϕ is of little influence, as it is the rotation around the z-axis: a rotation around the axis of symmetry of the camera is of relatively little influence for the error. It is smallest (is 0) when $\theta = \frac{\pi}{2}$ (in combination with $\psi = \frac{\pi}{2}$): the z-axis of projector and camera is at right angles. Indeed, this is the place where the intersection of the ray and the plane is geometrically best conditioned.

In practice, the system of equations 5.5 is solved with least squares: the third equation is also involved. Substituting the second and last equation in the third yields a similar expression for z_c. The corresponding error contribution for pixel deviations is also inversely proportional to the denominator of this expression (d_2). If this denominator is near 0 but the denominator of 5.6 (d_1) is not, or vice versa, there is no problem. Bad conditioning arises when both denominators are near 0, for example for $u_p = 0$:

$$\begin{aligned} d_2 &= u_c R_{21} + v_c R_{22} + f_c R_{23} \\ d_1 &= u_c R_{11} + v_c R_{12} + f_c R_{13} \\ \mathcal{E} &= d_1^2 + d_2^2 \approx 0 \end{aligned}$$

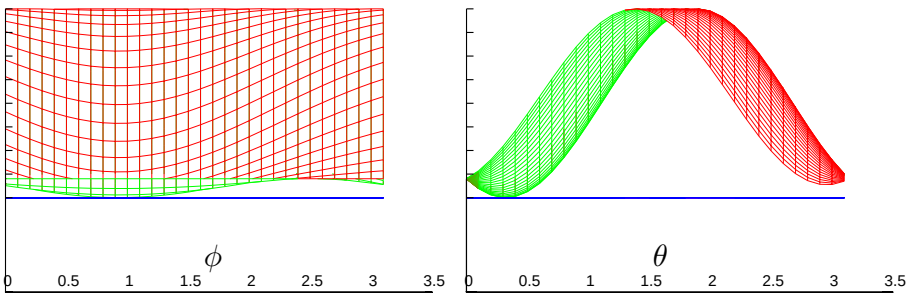


Figure 5.14: Side views of the function $\mathcal{E}$, sum of squared denominators

The right hand side of figure 5.13 shows $d_1^2+d_2^2$ for $u_c = 320\,pix$, $v_c = 240\,pix$ and $f_c = 1200\,pix$. Conditioning is worse as this function approaches 0, hence the plane $z = 0$ is also plotted. Figure 5.14 shows 2 side views: for this (u_c, v_c) pair $d_1^2 + d_2^2 = 0$ for $\phi \approx 0.9$, $\theta \approx 0.4$. $d_1^2 + d_2^2$ is not a function of ψ: the conditioning of both ray-plane intersections keep each other in balance, for $u_p = 0$:

$$\begin{aligned}\mathcal{E} =&[(2\cos(\phi)\sin(\phi)\cos(\theta)^2 - 2\cos(\phi)\sin(\phi))u_c - 2f_c\cos(\phi)\cos(\theta)\sin(\theta)]v_c \\ &+ (\cos(\phi)^2\cos(\theta)^2 + \sin(\phi)^2)v_c^2 + (\sin(\phi)^2\cos(\theta)^2 + \cos(\phi)^2)u_c^2 \\ &- 2f_c\sin(\phi)\cos(\theta)\sin(\theta)u_c + f_c^2\sin(\theta)^2\end{aligned} \tag{5.10}$$

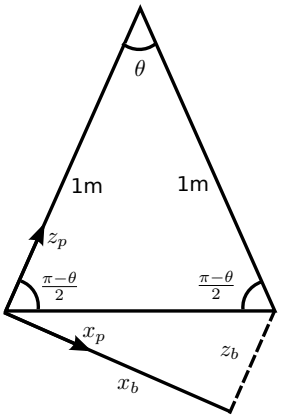


FIGURE 5.15: Assumption of stereo setup for numerical example

Now a numerical example of a realistic average error. Suppose the scene is at $z_c = 1m$. Assume one can locate the projected blob centres up to 2 pixels (with VGA resolution), then the average measurement error is $\Delta u_c = \Delta v_c = 1pix$. Most experiments are done with a AVT Marlin Guppy. The principal distance resulting from camera calibration with this camera is $f_c \approx 1200pix$. Let $y_b = 0$ and then suppose camera, projector and point to be reconstructed form a isosceles triangle (with sides of $1m$), see figure 5.15, then $x_b = \sin(\theta)m$. Let $\theta = \dfrac{\pi}{4}$ (the two imaging devices are at 45°, an average between good and bad conditioning), then $x_b = 0.71m$. Also for ψ choose a value that is in between good and bad conditioning: $\psi = \dfrac{\pi}{4}$. Then fill out equation 5.7 for these values

$$E_{coord} \approx (\frac{1m^2\ 1pix\ \sqrt{1-0.25}}{1200pix\ 0.71m})^2 \approx (1mm)^2$$

Thus the contribution of the error caused by an offset of 1 pixel in both directions in the camera is about $1mm$ on average. Later it will become clear that the contribution of this error is relatively limited compared to other contributions.

5.4.2.4 The contribution of errors in the intrinsic parameters

Contribution of camera principal distance To calculate the influence of the principal distance of the camera, consider a blob projected in the centre of the projection image to simplify the calculations: $u_p = 0$:

$$\frac{\partial z_c}{\partial f_c} = \frac{z_c^2(u_c R_{11} + v_c R_{12})}{f_c^2 x_b} = x_b \, \frac{u_c R_{11} + v_c R_{12}}{(u_c R_{11} + v_c R_{12} + f_c R_{13})^2} \tag{5.11}$$

Thus the error increases more than proportional with an increasing distance to the scene. It also increases with as baseline becomes wider: the entire setup scales up, and so does the error. The contribution to the overall error is 0 for the pixel in the centre of the image ($u_c = v_c = 0$). This is logical, as a change in principal distance changes the size of the pinhole model pyramid, but not its central point.

As a 4D cut cannot be visualised, choose $\psi = \frac{\pi}{2}$ again, for the same reason as in the previous paragraph: through the solution of system of equations 5.5 as an overdetermined system, the influence of a bad ray-plane conditioning for other values of ψ will be attenuated. Then:

$$E_{intr,f_c}(\psi = \frac{\pi}{2}) = (x_b \Delta f_c)^2 \, (\frac{u_c \cos(\theta)\sin(\phi) + v_c \cos(\theta)\cos(\phi)}{(u_c \cos(\theta)\sin(\phi) + v_c \cos(\theta)\cos(\phi) - f_c \sin(\theta))^2})^2 \tag{5.12}$$

This error function is determined by its denominator: when it approaches 0 the variations due to the numerator are negligible. Since the denominator is identical to the one of expression 5.9, the scaling of the error function in equation 5.12 is in good approximation the one depicted on the left hand side of figure 5.13. It represents a 3D cut of the square root of equation 5.12, where θ is limited to slightly larger values ($\theta > 0.5$). The error increases as θ approaches π (in combination with $\phi \approx \pi$): then the z-axes of the imaging devices are near parallel. For low θ the error also increases: again the z-axes are near parallel.

The remarks made in section 5.4.2.3 about the denominators of both z_c expressions not being 0 at the same time, are also valid here. Avoid $\mathcal{E} = 0$ with $\mathcal{E}$ as defined in equation 5.10.

For a numerical example, take $f_c = 1200\ pix$ as before. The worst case scenario is for camera image coordinates the furthest away from the centre, hence take $u_c = 320\ pix$, $v_c = 240\ pix$ (VGA resolution). Assume one makes a $\approx 1\%$ error: $\Delta f_c = 10\ pix$. Assume a isosceles triangle: x_b then depends on θ, $x_b = sin(\theta)$, see figure 5.15. The only variables that remain to be determined are the three Euler angles. Choose a value for ψ in between good and bad conditioning: $\psi = \frac{\pi}{4}$. For example, for $\theta = \frac{\pi}{4}$ equation 5.11 is only dependent on ϕ. Figure 5.16 shows the error contribution as a function of ϕ.

$$E_{intr,f_c} = (\frac{(u_c(0.7\cos(\phi) - 0.5\sin(\phi)) - v_c(0.7\sin(\phi) + 0.5\cos(\phi)))0.7 \cdot 10}{(u_c(0.7\cos(\phi) - 0.5\sin(\phi)) - v_c(0.7\sin(\phi) + 0.5\cos(\phi)) + \frac{1200}{2})^2})^2$$

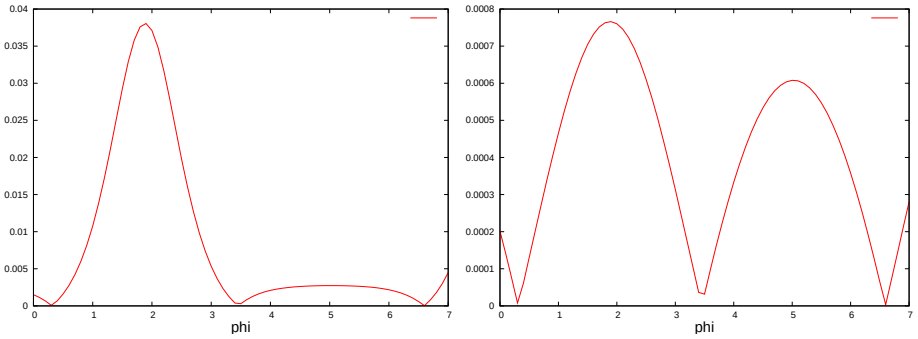


FIGURE 5.16: Error scaling with $\psi = \theta = \frac{\pi}{4}$: on the left for $u_c = 320\ pix$, $v_c = 240\ pix$, on the right for $u_c = 32\ pix$, $v_c = 24\ pix$

Hence, for the most distant pixel and the least favourable the error contribution is $\approx 4cm$, at $u_c = 32\ pix$, $v_c = 24\ pix$ the corresponding maximal error is less than a mm.

Contribution of projector principal distance If $u_p = 0\ pix$ then $\frac{\partial z_c}{\partial f_p}$ is also 0: an error in projector principal distance does not change the position of the central point in the projection image. As the change is linear with the distance from this central point, the worst case scenario is $u_p = (u_p)_{max}$. Suppose the projected blob is observed in the centre of the camera image ($u_c = v_c = 0\ pix$): this is as good as any point, but makes the formula simpler. Then:

$$\frac{\partial z_c}{\partial f_p} = \frac{z_c^2 u_p (R_{13} - x_b R_{33})}{(z_b u_p - x_b f_p)^2} = \frac{u_p (z_b \sin(\theta)\sin(\psi) - x_b \cos(\theta))}{(f_p \sin(\theta)\sin(\psi) - u_p \cos(\theta))^2}$$

The first equality indicates that this error increases again with increasing scene depth. The second equality expresses the partial derivative in only the base variables: when the denominator becomes ≈ 0, the error increases drastically. The left part of figure 5.17 plots the denominator and a $z = 0$ plane: the intersection defines a 2D curve of combinations of angles that need to be avoided. The right hand side of the figure depicts the evolution of the absolute value of the partial derivative outside the zone where the denominator makes the errors increase. The two peaks on the left are due the denominator approaching 0, the error increases as theta approaches $\frac{\pi}{2}$.

To give realistic values to the baseline vector, assume a isosceles triangle again (figure 5.15), then $z_b = 2[\cos(\frac{\pi - \theta}{2})]^2 m$, $x_b = \sin(\theta) m$:

$$E_{intr,f_p} = (u_p \Delta f_p \cdot \frac{2[\cos(\frac{\pi-\theta}{2})]^2 \sin(\theta)\sin(\psi) - \sin(\theta)\cos(\theta)}{(f_p \sin(\theta)\sin(\psi) - u_p \cos(\theta))^2})^2 \qquad (5.13)$$

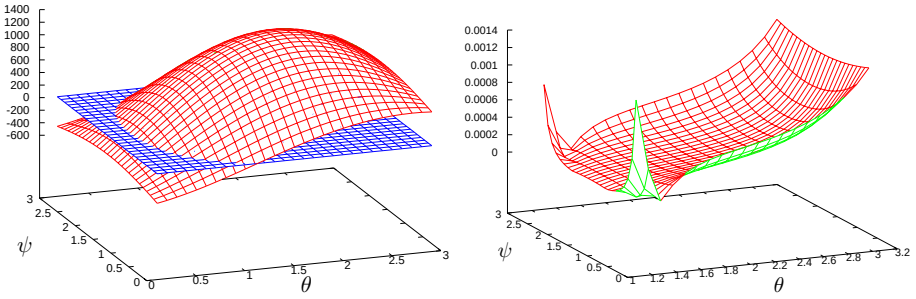


FIGURE 5.17: Left: denominator of partial derivative, right: absolute value of the partial derivative outside the zone where the denominator approaches 0

Only two parameters remain: figure 5.18 shows this 3D cut as a function of ψ and θ.

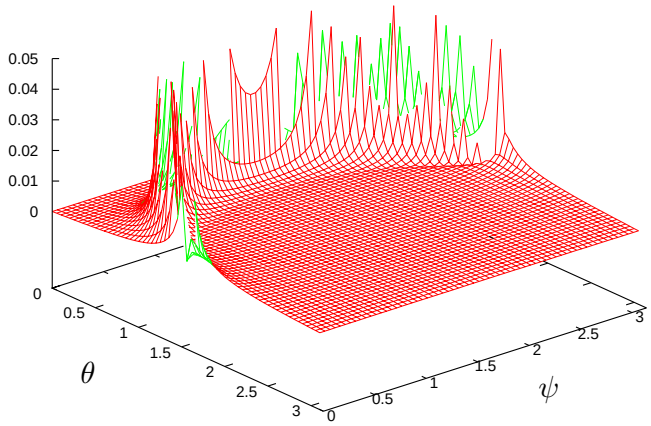


FIGURE 5.18: Cut of the second factor of equation 5.13 in function of ψ and θ

For a numerical example assume a XGA projector resolution: $u_p = 512\ pix$. Projector calibration – for the projector used in the experiments – yields a principal distance of $\pm 2000\ pix$. Assume a 1% error: $\Delta f_p = 20\ pix$.

Previously errors have been calculated for $\psi = \theta = \frac{\pi}{4}$: this point is near the curve of points that make the denominator in equation 5.13 0, in the area that surpasses the maximum value of the z-axis in figure 5.18. Using these values $E_{intr,f_p} = (2cm)^2$ (near worst case scenario). For example for $\psi = \frac{\pi}{2}$, with the same θ (better ray-plane intersection): $E_{intr,f_p} = (2mm)^2$.

5.4.2.5 The contribution of errors in the extrinsic parameters

Contribution of the baseline

$\mathbf{x_b}$) For a blob in the centre of the projector

$$\frac{\partial z_c}{\partial x_b} = \frac{z_c}{x_b} \Rightarrow E_{extr,x_b} = (\frac{z_c \Delta x_b}{x_b})^2 \tag{5.14}$$

Hence, the further away the scene, the larger the error; the larger the baseline, the smaller the influence for a deviation in that baseline. For example, consider the case with $\theta = \frac{\pi}{4}$ with $z_c = 1m$. Then for $\Delta x_b = 1cm$, and for a isosceles triangle ($x_b = 0.71m$) $E_{extr,x_b} = (1.4cm)^2$. A considerable error for a margin of about 1% in the baseline coordinate. Equation 5.14 contains a $z_c(\theta, \psi, \phi, \dots)$ in the numerator. Substituting yields the same numerator as equation 5.8. Hence the error becomes infinite for the same combinations of angles as put forward in section 5.4.2.3.

$\mathbf{z_b}$) This remark is also valid for the contribution of the other baseline coordinate, z_b, as it also has z_c in the numerator:

$$\frac{\partial z_c}{\partial z_b} = \frac{u_p z_c}{z_b u_p - x_b f_p} \Rightarrow E_{extr,z_b} = (\frac{u_p z_c \Delta z_b}{z_b u_p - x_b f_p})^2$$

(x_b, y_b, z_b) is expressed in the projector frame: z_b is measured parallel to the z_p axis. Hence there is no contribution due to Δz_b when for a blob in the centre of the projection image, indeed $u_p = 0 \Rightarrow E_{extr,z_b} = 0$. Worst case scenario for this error is when $u_p = (u_p)_{max}$. A numerical example for the same case with $\theta = \frac{\pi}{4} \Rightarrow z_b = 0.29m$ and $\Delta z_b = 1cm$:

$$E_{extr,z_b} = (\frac{512pix\ 1m\ 0.01m}{0.29m\ 512pix - 0.71m\ 2000pix})^2 = (4mm)^2$$

Contribution of the frame rotation

$\boldsymbol{\psi}$) To simplify the equations, consider the case of $\psi = \frac{\pi}{2}$, for the same reasons as above: this error calculation is only based on 3 of the 4 equations of the system of equation 5.5, the least squares solution of the entire overdetermined system will minimise the possible bad conditioning of the ray-plane intersection by adding another plane equation. Then for a blob in the centre of the projection image ($u_p = 0\ pix$):

$$\frac{\partial z_c}{\partial \psi}(\psi = \frac{\pi}{2}) = \frac{f_c x_b (u_c \cos(\phi) - v_c \sin(\phi))}{(u_c \cos(\theta)\sin(\phi) + v_c \cos(\theta)\cos(\phi) - f_c \sin(\theta))^2} \tag{5.15}$$

The corresponding error function is determined by its denominator, which becomes 0 at the intersection of the two surfaces at the right hand side of figure

5.12. Therefore, the error contribution scales in function of ϕ and θ as in the left hand side of figure 5.13. Figure 5.19 plots this function for the full range of θ and ϕ from 0 to π.

According to equation 5.15, for $\psi = \frac{\pi}{2}$ the error contribution for the principal point is 0, logical: when the ray-plane intersection is perfect, the error increment is minimal. For an average error take $\psi = \frac{\pi}{4}$ (between good and bad conditioning), then for a point in the centre of the camera image:

$$E_{extr,\psi}(u_c = v_c = 0pix) = (\frac{\sqrt{2}x_b\Delta\psi}{\sin(\theta)})^2 \tag{5.16}$$

Thus ideally $\theta = \frac{\pi}{2}$, worst case scenario: $\lim_{\theta\to 0} E_{extr,\psi} = \lim_{\theta\to\pi} E_{extr,\psi} = \infty$. If the point is not in the centre of the image, the locations where the error becomes infinite are slightly influenced by ϕ, as can be seen in figure 5.19.

For a realistic $\theta = \frac{\pi}{4}$, $\Delta\psi = \frac{\pi}{100}$, $x_b = 0.71m$ ($z_c = 1m$): $E_{extr,\psi} = (4cm)^2$. Hence, if one would only use 3 out of 4 equations this would be a sensitive parameter: a deviation of $\approx 2^\circ$ gives rise to an error of several centimetres. Therefore, it is wise to take all equations into account.

θ) To calculate the contribution due to the angle θ, also consider the case of $\psi = \frac{\pi}{2}$ (for the same reasons as above). Considering a feature in the centre of the projection image:

$$\frac{\partial z_c}{\partial\theta}(\psi = \frac{\pi}{2}) = -f_c x_b \frac{u_c\sin(\phi)\sin(\theta) + v_c\cos(\phi)\sin(\theta) + f_c\cos(\theta)}{(u_c\sin(\phi)\cos(\theta) + v_c\cos(\phi)\cos(\theta) - f_c\sin(\theta))^2} \tag{5.17}$$

Thus for a point in the centre of the image:

$$E_{extr,\theta}(u_c = v_c = 0pix) = (\frac{x_b\cos(\theta)\Delta\theta}{[\sin(\theta)]^2})^2 \tag{5.18}$$

All remarks about the denominators of equations 5.15 and 5.16, are also valid there for equations 5.17 and 5.18.
A numerical example for the principal point of the image for an angular error of $\frac{\pi}{100}$, under the isosceles triangle assumption: $E_{extr,\theta}(\psi = \theta = \frac{\pi}{4}) = (\sqrt{2}x_b\Delta\theta)^2 = (3cm)^2$, again a deviation of $\Delta\theta \approx 2^\circ$ contributes several cm error.

φ) For the contribution of the angle ϕ, consider $\psi = \frac{\pi}{2}$ again:

$$\frac{\partial z_c}{\partial\phi}(\psi = \frac{\pi}{2}) = f_c x_b \frac{\cos(\theta)(u_c\cos(\phi) - v_c\sin(\phi))}{(u_c\sin(\phi)\cos(\theta) + v_c\cos(\phi)\cos(\theta) - f_c\sin(\theta))^2}$$

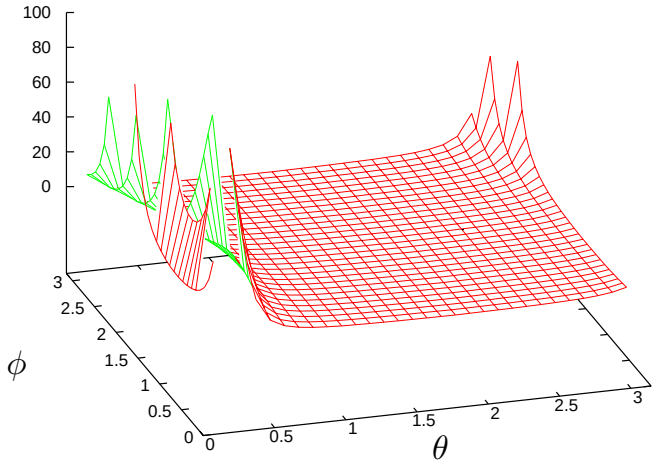


FIGURE 5.19: plot of $|\frac{\partial z_c}{\partial \theta}|$, similar plots for $|\frac{\partial z_c}{\partial \phi}|$ and $|\frac{\partial z_c}{\partial \psi}|$

Thus the error contribution is 0 for the principal point of the camera. This is logical, as ϕ represents a rotation around the axis of rotation of the camera: an error in that rotation does only influence pixels away from the central pixel. For those pixels, the error contribution is again largely determined by the denominator approaching 0. This denominator is the same as in equation 5.15 and 5.17, hence the resulting plot is similar to the plot in figure 5.19: the contributions are only large for θ near 0 or π (then the z-axes of the imaging devices are near parallel).

For a numerical example, consider $\psi = \theta = \frac{\pi}{4}$, then:

$$E_{extr,\phi}(\psi = \theta = \frac{\pi}{4}) = (\frac{f_c x_b[u_c(\sqrt{2}\sin(\phi) + \cos(\phi)) + v_c(\sqrt{2}\cos(\phi) - \sin(\phi))]\Delta\phi}{[u_c(\cos(\phi) - \frac{\sin(\phi)}{\sqrt{2}}) - v_c(\sin(\phi) + \frac{\cos(\phi)}{\sqrt{2}}) + \frac{f_c}{\sqrt{2}}]^2})^2$$

Figure 5.20 plots this error in function of ϕ for $u_c = (u_c)_{max} = 320\ pix$, $v_c = (v_c)_{max} = 240\ pix$: a deviation of $\approx 2^\circ$ can lead to an error of several cm.

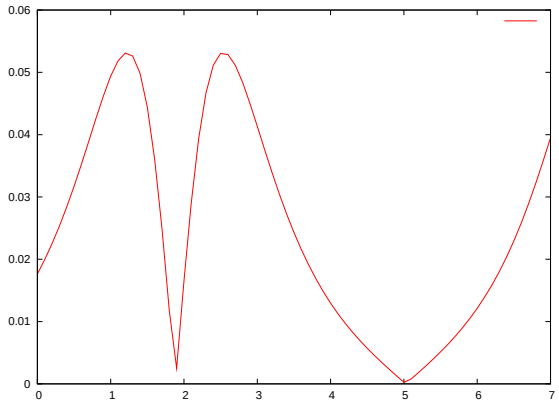


FIGURE 5.20: $\sqrt{E_{extr,\phi}(\psi = \theta = \frac{\pi}{4})}$ as a function of ϕ

5.5 Conclusion

This section studied the influence of each of the measurements and calibration parameters on the accuracy of the 3D reconstruction. The contribution of the measurements, pixel deviations, is typically in the order of a few mm. It is relatively limited compared to the errors caused by erroneous calibration. The intrinsic parameters for example: a deviation in the principal distance is small for pixels near the principal point, but can cause errors of several cm near the edges. Of the extrinsic parameters the influence of the baseline is typically around $1cm$, the rotational parameters are more sensitive: up to several cm. Thus in studying the error propagation, the ones caused by imperfect calibration are not negligible.

However, the reconstruction equations have singularities (denominators that can become 0). Fortunately, the system of equations is overdetermined and these singularities liquidate each other partly. These singularities are dependent on u_c, v_c, f_c, and the Euler angles θ and ϕ but independent of ψ. Approximately, it says that θ cannot be close to 0 or π, logically, then the z-axes of the imaging devices are near parallel. ϕ is of lesser influence, as it is the rotation of the camera around its axis of rotation.

Fortunately, as the aim in robotics is not to precisely reconstruct a scene, but to navigate visually, the accuracy is sufficient: it is constantly improved as the robot moves closer to the object of interest. The accuracy of the Cartesian positioning of the robot arm itself is of less importance, as the control does not need to be accurate enough with respect to the robot base frame, but with respect to the object frame.

Chapter 6

Robot control

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

This chapter studies the control of the robotic arm and hardware related issues of the robot and its vision sensor. Section 6.1 explains the hardware used for the sensor described in the previous chapters. Section 6.2 discusses the kinematics of the robot.

6.1 Sensor hardware

Figure 6.1 shows the hardware involved in the setup, the arrows indicate the communication directions. We use a standard PC architecture. Three devices are connected to the control PC: camera, projector and robot. Each of them has a corresponding software component.

6.1.1 Camera

A possible camera technology is a frame grabber system: it performs the A/D conversion on a dedicated PCB. This thesis does not use such system, as it is more expensive than other systems and provides no standard API. All systems discussed below do the A/D conversion on the camera side.

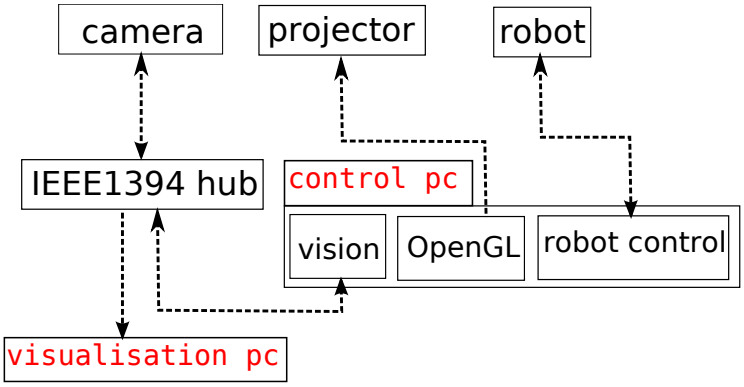


Figure 6.1: Overview of the hardware setup

FireWire

This work chooses a IEEE1394a interface, as its protocol stack provides the DCAM (IIDC) camera standard. This *FireWire* protocol provides not only a standardised hardware interface, but also standardised software access to camera functions for any available camera. Alternative standards that have a comparable bandwidth like USB2 ($480Mbps$ vs $400Mbps$ for IEEE1394a), do not provide such standardised protocol: they require a separate driver and corresponding control functions for each type of camera. Note that the DCAM standard only applies to cameras that transmit uncompressed image data like webcams and industrial cameras (as opposed to e.g. digital camcorders).

Control frequency The image frame rate f_r is important for smooth robot control. It is limited by the bandwidth Δf of the channel. Section 7.3 explains which factors the bandwidth is composed of: $\Delta f \sim$ resolution $\cdot f_r$. Thus, if the resolution can be decreased because a full resolution does not contribute to the robot task, the frame rate can be increased. Another possibility is to select only a part of the image at a constant resolution. Some of these *FireWire* cameras [1] — both CMOS and CCD based ones — allow the selection of a **r**egion **o**f **i**nterest. If this features is available, it can be controlled through the standard software interface. This way, the frequency can be increased to several hundred Hz. The new limiting factor then becomes the shutter speed of the camera. The inverse of this increased frame rate needs to remain larger than the integration time needed to acquire an image that is bright enough. Fortunately, as the structured light system works with a very bright light source, this integration time can be limited to a minimum, and high frequencies are possible.

[1] the AVT Guppy for example

GigE Vision

The newer gigabit Ethernet interface however does have a standardised interface similar to DCAM (called *GigE Vision*), offers more bandwidth (1000Mbps) and a longer cable length. The former is not an asset for this application as we need only one camera, and resolutions higher than VGA will not improve system performance. The latter however is an advantage, as IEEE1394a cables are limited to $4.5m$ (without daisy chaining using repeaters or hubs). The GigE standard uses UDP, since the use of TCP at the transportation layer would introduce unacceptable image latency. UDP is a reasonable choice as it is not essential that every image is transferred perfectly.

Computational load

CPU benchmarks learn that while capturing images half of CPU time is consumed by transferring the images, and half by their visualisation. Therefore it may be interesting to use a second PC for the visualisation to reduce the load on the control PC: a *FireWire* hub copies the camera signal and lead it to a visualisation PC.
Almost all industrial IIDC-compliant cameras use a **Bayer filter** in front of the image sensor, reducing to actual resolution of both image dimensions by 50% for the green channel, 25% for the red channel and 25% for the blue channel. Therefore the green channel has only a fourth of the pixels of the indicated resolution, the red and the blue channel only a sixteenth.

Interpolating the missing information with a demosaicing algorithm needs to be done in software, and hence by the CPU, and is a considerable load. Some more expensive camcorders use a 3CCD system that splits the light by a trichroic prism assembly, which directs the appropriate wavelength ranges of light to their respective CCDs, and thus work at the native resolution in all colour channels.
Let v be 0 or 1 whether or not visualisation is required, and d 0 or 1 if debayering in software is required, then an approximate empirical formula for the CPU load while capturing images is

$$\text{load} \ \sim fps \ (1+v) \ (1+5d)$$

However if we use the resolution of the green channel as it is, and interpolate the red and blue channels to the size of the green channel (resulting in 25% of the pixels of the original image) the CPU load is only about

$$\text{load} \ \sim fps \ (1+v) \ (1+0.8d)$$

Indeed: 6 times less work debayering: let l be the work for the smaller image, then the work for one channel in this image is $\frac{l}{2}$. The larger image has 4 times the pixels, and 3 channels, so the work for the larger image is $\frac{4 \cdot 3w}{2}$. Since the full resolution image is only an upsampled interpolated version of this measured image, this is the correct way to work: working at full resolution is simply a

waste of CPU cycles. Clearly, if the camera is only to be used to reconstruct parts of the scene using grayscale features, a grayscale camera suffices, and the above is not relevant. However, the colour information could for example be used in 2D vision that is combined with the 3D vision, as in section 8.4.5.

6.1.2 Projector

Demands for the projector are rather simple, and on the lower end of the 2007 consumer market. SVGA (800×600) or XVGA (1024×768) resolution suffices, and so does a brightness output of 1500 lumen. However, it must be able to focus at relatively close range ($\approx 1m$), an unusual demand as most projectors are designed to focus on a screen several meters away. One could change the lens of the projector, but this is not necessary, as for example the *Nec VT57* is a relatively cheap (700$ in 2007) projector that can produce a sharp image at relatively short range: at a distance of $70cm$. Mind that automatic geometric corrections such as the *Keystone* correction are disabled, otherwise the geometry model is clearly not valid.

6.1.3 Robot

Since the standard controllers for robot arms currently commercially available do not provide primitives for adequate sensor-based control, these controllers have been bypassed and replaced by a normal PC with digital and analogue multichannel I/O PCI cards, and more adapted software. Our group uses the *Orocos* [Bruyninckx, 2001] software (see section 7.2.4): it provides a uniform interface to all robots in its library and their proprioceptive sensors, often a rotational encoder for every joint. *Orocos* has a real-time mode, for which it is based on a real time operating system. Many authors use the term "real-time" when they intend to say "online", this thesis defines the term as having operational deadlines from event to system response: the system has to function within specified time limits.

6.2 Motion control

6.2.1 Introduction

Section 5.4.2 studied the sensitivity of the reconstruction equations in terms of the $z-x-z$ convention Euler angles. The rotational part of a twist however is expressed in terms of the angular velocity $\boldsymbol{\omega}$. The relationship between both 3D vectors is linear: the *integrating factor* $\mathbf{E}$ (dependent on the Euler angle convention) relates them:

$$\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix}\mathbf{t} = \boldsymbol{\omega} = \mathbf{E}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \text{ with } \mathbf{E} = \begin{bmatrix}0 & \cos(\phi) & \sin(\phi)\sin(\theta)\\ 0 & \sin(\phi) & -\cos(\phi)\sin(\theta)\\ 1 & 0 & \cos(\theta)\end{bmatrix} \tag{6.1}$$

For every combination of joint positions $\mathbf{q}$, we calculate the robot Jacobian $\mathbf{J}$ such that from a desired twist (6D velocity) $\mathbf{t}$ we can calculate joint velocities $\dot{\mathbf{q}} \equiv \dfrac{\partial \mathbf{q}}{\partial t}$. To this end, we use the robotics software *Orocos* developed in our group, see section 7.2.4. With n the number of degrees of freedom of the robot:

$$\mathbf{t}_i = \mathbf{J}_{R,i}\dot{\mathbf{q}} \Leftrightarrow \begin{bmatrix}\frac{\partial x_i}{\partial t}\\ \frac{\partial y_i}{\partial t}\\ \frac{\partial z_i}{\partial t}\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}\boldsymbol{I} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{E}\end{bmatrix}\begin{bmatrix}\frac{\partial x_i}{\partial q_1} & \frac{\partial x_i}{\partial q_2}\cdots & \frac{\partial x_i}{\partial q_n}\\ \frac{\partial y_i}{\partial q_1} & \frac{\partial y_i}{\partial q_2}\cdots & \frac{\partial y_i}{\partial q_n}\\ \frac{\partial z_i}{\partial q_1} & \frac{\partial z_i}{\partial q_2}\cdots & \frac{\partial z_i}{\partial q_n}\\ \frac{\partial \phi_i}{\partial q_1} & \frac{\partial \phi_i}{\partial q_2}\cdots & \frac{\partial \phi_i}{\partial q_n}\\ \frac{\partial \theta_i}{\partial q_1} & \frac{\partial \theta_i}{\partial q_2}\cdots & \frac{\partial \theta_i}{\partial q_n}\\ \frac{\partial \psi_i}{\partial q_1} & \frac{\partial \psi_i}{\partial q_2}\cdots & \frac{\partial \psi_i}{\partial q_n}\end{bmatrix}\begin{bmatrix}\frac{\partial q_1}{\partial t}\\ \frac{\partial q_2}{\partial t}\\ \cdots\\ \frac{\partial q_n}{\partial t}\end{bmatrix} \tag{6.2}$$

$\mathbf{t}$ and $\mathbf{J}$ are dependent on the reference frame considered (assume the reference point is in the origin of the reference frame), hence the subscript i in equation 6.2: it is to be replaced by a specific frame indicator. Note that the frame transformation from the camera frame to the end effector frame needs to be accounted for here, see figure 6.2. This is done using the results of the hand-eye calibration of section 4.4.

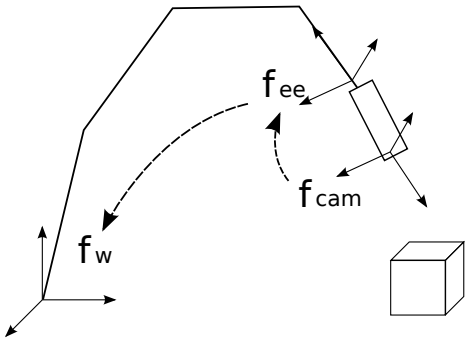


Figure 6.2: Frame transformations: world frame, end effector frame and camera frame

6.2.2 Frame transformations

Let f_1 be a frame in which one wants to express the desired twist, and f_3 a frame in which one can easily express it. f_1 and f_3 generally differ both in translation and rotation.

Expressing t for a frame translation

Let $\mathbf{p}$ be the displacement vector between the two frames f_1 and f_2 with the same angular orientation, and their respective twists $\mathbf{t}_{f1}$ and $\mathbf{t}_{f2}$. Figure 6.3 is an example of such a situation. The next section will deal with this example more extensively. Then $\boldsymbol{v_{f2} = v_{f1} + p \times \omega_{f_1}}$. Define the 6×6 matrix

$$\mathbf{M}_{f2}^{f1} = \begin{bmatrix} \mathbf{I} & [\mathbf{p}]_\times \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \Rightarrow \mathbf{t}_{f2} = \mathbf{M}_{f2}^{f1} \mathbf{t}_{f1}$$

In rigid body kinematics the superscript indicates the reference frame, the subscript the destination frame: for example $\mathbf{M}_{f2}^{f1}$ describes the translational transformation from f_1 to f_2.

Expressing t for a frame rotation

Consider two frames f_2 and f_3 only different in rotation and not in translation: the reference point remains invariant. Express the twist in frame f_3: define the 6×6 matrix $\mathbf{P}_{f3}^{f2} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$ with $\mathbf{R}$ the 3×3 rotation matrix to rotate from f_2 to f_3, then $\mathbf{t}_{f3} = \mathbf{P}_{f3}^{f2} \mathbf{t}_{f2}$.

Frame rotation and translation combined

Define the screw transformation matrix

$$\mathbf{S}_{f3}^{f1} = \mathbf{P}_{f3}^{f2} \mathbf{M}_{f2}^{f1} = \begin{bmatrix} \mathbf{R} & \mathbf{R}[\mathbf{p}]_\times \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \Rightarrow$$

$\mathbf{S}_{f3}^{f1} \mathbf{J}_{R,f1} \dot{\mathbf{q}}$ is equal to a twist in frame f_3 in which the desired twist can easily be expressed.

6.2.3 Constraint based task specification

Our lab [De Schutter et al., 2007] developed a systematic constraint-based approach to execute robot tasks for general sensor-based robots consisting of rigid links. It is a mathematically clean, structured way of combining different kinds of sensory input for instantaneous task specification. This combination is done using weights that determine the relative importance of the different sensors. If the combined information does not provide enough constraints for an exactly determined robot motion, the extra degrees of freedom are filled out using some general limitation, like minimum kinetic energy. If the constraints of the different sensors are in violation with each other, the weights determine what specifications will get the priority and which will not.
The remainder of this section demonstrates this technique in two case studies using computer vision. The first one is a 3D positioning task, using 2D vision. The second one is for any 6D robot task, using 3D – stereo – vision, in this case structured light.

Case study 1: 3D positioning task

Consider the setup of figure 6.3. On the left you see a rectangular plate attached to the end effector. This plate needs to be aligned with a similar plate on a table. This experiment is a simplification of the task to fit car windows in the vehicle body. One could preprogramme this task to be executed repetitively with the exact same setup, but then reprogramming is necessary when the dimensions of either window or vehicle body change. Or one could use a camera in an eye-to-hand setup to fit both objects together as in this experiment. The right hand side of figure 6.3 shows a top view of the plate held by the object and the plate on the table, with the indication of a minimal set of parameters to define their relative planar orientation: a 3DOF problem: 2 translations d_1, d_2 and one rotation α. Consider the robot base frame f_1 to be the world frame. Consider now the closed kinematic loop $w \rightarrow ee \rightarrow f_4 \rightarrow f_3 \rightarrow f_2 \rightarrow w$. The relative twists are constrained by the *twist closure equation*, expressed in the world coordinate system:

$$
\begin{aligned}
{}_w\mathbf{t}_w^{ee} + {}_w\mathbf{t}_{ee}^{f4} + {}_w\mathbf{t}_{f4}^{f3} + {}_w\mathbf{t}_{f3}^{f2} + {}_w\mathbf{t}_{f2}^{w} &= \mathbf{0} \\
{}_w\mathbf{t}_w^{ee} + \mathbf{0} + {}_w\mathbf{t}_{f4}^{f3} + \mathbf{0} + \mathbf{0} &= \mathbf{0} \\
{}_w\mathbf{t}_w^{ee} &= {}_w\mathbf{t}_{f3}^{f4} \\
\mathbf{J}_{R,f1}\dot{\mathbf{q}} &= \mathbf{S}_{f1}^{f3} \cdot {}^d\mathbf{t}_{f3}^{f4} \\
\mathbf{S}_{f3}^{f1}\mathbf{J}_{R,f1}\dot{\mathbf{q}} &= {}^d\mathbf{t}_{f3}^{f4}
\end{aligned}
$$

where ${}_k\mathbf{t}_j^i$ represents the relative twist of frame i with respect to frame j, expressed in frame k. ${}^d\mathbf{t}_{f3}^{f4}$ is the desired twist of frame 4 with respect to frame 3, expressed in frame 3.

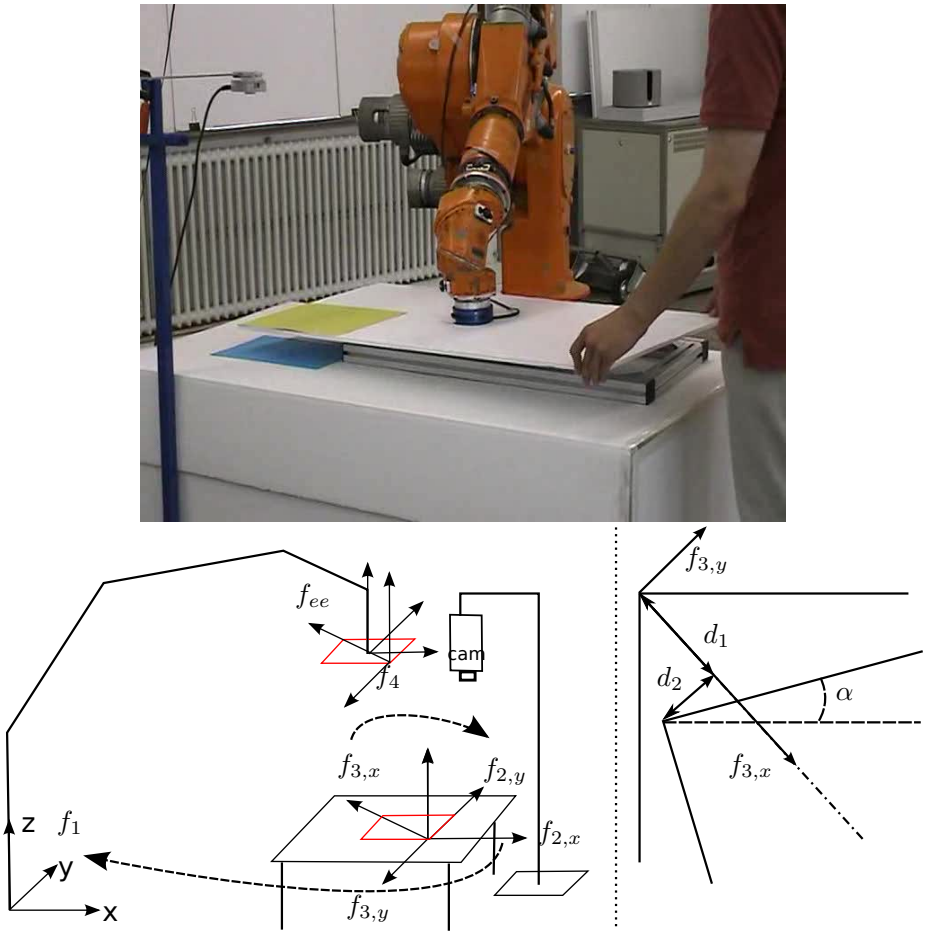


Figure 6.3: Top: lab setup, bottom left: choice of frames: translation from f_1 to f_2, rotation from f_2 to f_3; bottom right: 2D top view of the table and the object held by the end effector

This setup only puts constraints on 3 of the 6 DOF. The first two rows correspond to respectively the x and y axis of f_3, and are dimensions in which we want to impose constraints to the robot. The sixth row is the rotation around the z-axis of f_3: also a constraint for the robot. We assume a P-controller: the desired twist is proportional to the errors made: d_1, d_2 and α. As these parameters are not in the same parameter space, a scalar control factor will not be sufficient: let $\mathbf{K}$ be the matrix of feedback constants (typically a positive-definite diagonal matrix). Let $\mathbf{A}$ be the matrix formed by taking the first 2 and

the last row of $\mathbf{S}_{f3}^{f1}\mathbf{J}_{R,f1}$, then:

$$^{d}\begin{bmatrix} v_{x,f3} \\ v_{y,f3} \\ \omega_{z,f3} \end{bmatrix} = \mathbf{K}\begin{bmatrix} d_2 \\ d_1 \\ \alpha \end{bmatrix} \equiv \mathbf{Ke} = \mathbf{A}\dot{\mathbf{q}} \Rightarrow \dot{\mathbf{q}} = \mathbf{A}^{\#}\mathbf{Ke} \tag{6.3}$$

This is an underconstrained problem, therefore an infinite number of solutions exist. One needs to add extra constraints to end up with a single solution. One possible constraint is to minimise joint speeds. Choose $\mathbf{W}$ such that it minimises the weighted joint space norm

$$\|\dot{\mathbf{q}}\|_{\mathbf{W}} = \dot{\mathbf{q}}^T\mathbf{W}\dot{\mathbf{q}}$$

For example, if the relative weights of the joint space velocities are chosen to be the mass matrix of the robot, the corresponding solution $\dot{\mathbf{q}}$ minimises the kinetic energy of the robot.
$\mathbf{A}^{\#}$ minimises $\|\mathbf{A}\dot{\mathbf{q}} - \mathbf{Ke}\|_W$. This weighted pseudoinverse can be calculated as follows. From equation 6.3: $\mathbf{AW}^{-1}\mathbf{W}\dot{\mathbf{q}} = \mathbf{Ke}$. Let $\mathbf{s} = \mathbf{W}\dot{\mathbf{q}}$ then

$$\mathbf{s} = (\mathbf{AW}^{-1})^{\dagger}\mathbf{Ke}$$

As the rows of $\mathbf{A}$ are linearly independent (the columns are not), $\mathbf{AA}^T$ is invertible, and an explicit formula for the Moore-Penrose pseudo-inverse exists

$$(\mathbf{AW}^{-1})^{\dagger} = (\mathbf{AW}^{-1})^T((\mathbf{AW}^{-1})(\mathbf{AW}^{-1})^T)^{-1} \Rightarrow$$

$$\dot{\mathbf{q}} = \mathbf{W}^{-1}(\mathbf{W}^{-1})^T\mathbf{A}^T(\mathbf{AW}^{-1}(\mathbf{W}^{-1})^T\mathbf{A}^T)^{-1}\mathbf{Ke}$$

Since $\mathbf{W}$ is diagonal (and thus also easily invertible):

$$\mathbf{W} = \mathbf{W}^T \Rightarrow \dot{\mathbf{q}} = \mathbf{W}^{-2}\mathbf{A}^T(\mathbf{AW}^{-2}\mathbf{A}^T)^{-1}\mathbf{Ke}$$

For this case study it may be important that both planes (the one that has f_4 attached, and the one that has f_3 attached) remain parallel. This may be more important than the minimisation of the joint speeds. In that case, remove the constraint that minimises the weighted joint space norm, and add the constraints $\omega_{x,f3} = 0$ and $\omega_{y,f3} = 0$.

Case study 2: 6D task using structured light

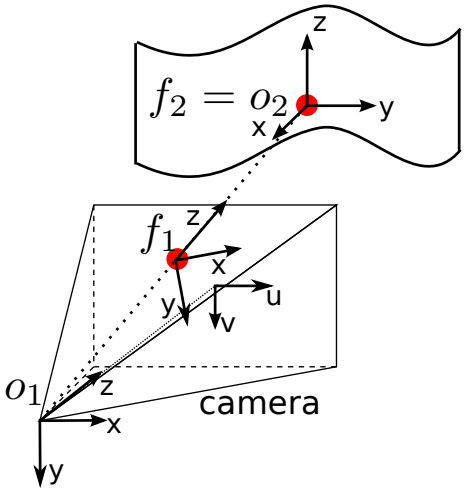


FIGURE 6.4: Frame transformations: object and feature frames

This section explains how to **position** the camera attached to the end effector **with respect to the projected blobs** in the scene. As a projector can be seen as an inverse camera, the same technique is applicable to the projector. This technique applies *constraint based task specification* to the structured light application studied throughout this thesis. It uses more of the mathematical elements of the theory than the previous case study, since in this case there are also points of interest in the scene that are not rigidly attached to objects. The position of a projected blob on the scene for instance, cannot be rigidly attached to an object. Then *constraint based task specification* defines two object and two feature frames for this task relation. Object frames are frames that are rigidly attached to objects, feature frames are linked to the objects but not necessarily rigidly attached to them. Figure 6.4 shows this situation: the object frame o_1 is rigidly attached to the projector with the z-axis through the centre of the pinhole model and the principal point. The object frame o_2 is rigidly attached to an object in the scene. Feature frame f_1 is the intersection of the ray with the image plane of the pinhole model of the camera, its z-axis is along the incident ray. Feature frame f_2 is attached to the point where the ray hits the scene.

The submotions between o_1 and o_2 are defined by the (instantaneous) twists $\mathbf{t}_{o1}^{f1}$, $\mathbf{t}_{f1}^{f2}$ and $\mathbf{t}_{f2}^{o2}$. Since the six degrees of freedom between o_1 and o_2 are distributed over the submotions, these twists respectively belong to subspaces of dimension n_1, n_2 and n_3, with $n_1 + n_2 + n_3 = 6$. For instance, for figure 6.4 there are two degrees of freedom between o_1 and f_1 ($n_1 = 2$: u and v), and four between f_1 and f_2 ($n_2 = 4$: one translation along the z-axis of f_1, and 3 rotational parameters). As this thesis is limited to estimating points and does not recognise objects, the object 2 frame is coincident with the feature 2 frame: $n_3 = 0$.

Twists $\mathbf{t}_{o1}^{f1}$, $\mathbf{t}_{f1}^{f2}$ and $\mathbf{t}_{f2}^{o2}$ are parametrised as a set of *feature twist coordinates* $\boldsymbol{\tau}$. They are related to the twists $\mathbf{t}$ through equations of the form $\mathbf{t} = \mathbf{J}_{Fi}\boldsymbol{\tau_i}$, with $\mathbf{J}_{Fi}$ the feature Jacobian of dimension $6 \times n_i$, and $\boldsymbol{\tau_i}$ of dimension $n_i \times 1$.

For example, for a ray of structured light (see figure 6.4), then $\mathbf{t}_{f2}^{o2} = \mathbf{0}$ and the origin of f_2 is along the z-axis of f_1:

$$ {}_{f1}\mathbf{t}_{f1}^{f2} = \begin{bmatrix} \mathbf{0}_{2\times 4} \\ \mathbf{I}_{4\times 4} \end{bmatrix} \begin{bmatrix} \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} $$

Neglecting a non-zero principal point, the u and v axes have the same orientation as the x and y axes of frame o_1, see the lower side of figure 4.5. Hence, the rotation from o_1 to f_1 is a rotation with $z - x - z$ convention Euler angles $\phi = -\arctan(\frac{u}{v})$, $\theta = \arctan(\frac{\sqrt{u^2+v^2}}{f})$ and $\psi = 0$. The homogeneous transformation matrix $\mathbf{T}_{o1}^{f1}$ expresses the relative pose of frame f_1 with respect to frame o_1:

$$ \mathbf{T}_{o1}^{f1} = \begin{bmatrix} \mathbf{R}_{o1}^{f1} & \mathbf{p}_{o1}^{f1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \text{ with } \mathbf{p}_{o1}^{f1} = [u\ v\ f]^T \text{ and } \mathbf{R}_{o1}^{f1} = \begin{bmatrix} \frac{v}{\beta} & -\frac{u}{\beta} & 0 \\ \frac{uf}{\alpha\beta} & \frac{vf}{\alpha\beta} & -\frac{\beta}{\alpha} \\ \frac{u}{\alpha} & \frac{v}{\alpha} & \frac{f}{\alpha} \end{bmatrix} $$

with $\alpha = \sqrt{u^2+v^2+f^2}$ and $\beta = \sqrt{u^2+v^2}$. $\mathbf{p}$ can be scaled arbitrarily (the size of the pinhole model is unimportant to the robot task). Use the integrating factor $\mathbf{E}$ to express $\mathbf{t}_{o1}^{f1}$ (see equation 6.1):

$$ \begin{aligned} {}_{o1}\mathbf{t}_{o1}^{f1} &= \mathbf{E} \begin{bmatrix} \frac{\partial p_x}{\partial u} & \frac{\partial p_y}{\partial u} & \frac{\partial p_z}{\partial u} & \frac{\partial \phi}{\partial u} & \frac{\partial \theta}{\partial u} & \frac{\partial \psi}{\partial u} \\ \frac{\partial p_x}{\partial v} & \frac{\partial p_y}{\partial v} & \frac{\partial p_z}{\partial v} & \frac{\partial \phi}{\partial v} & \frac{\partial \theta}{\partial v} & \frac{\partial \psi}{\partial v} \end{bmatrix}^T \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & \frac{-u^2 f}{\alpha^2\beta^2} & \frac{uvf}{\alpha^2\beta^2} & \frac{v}{\beta^2} \\ 0 & 1 & 0 & \frac{-uvf}{\alpha^2\beta^2} & \frac{v^2 f}{\alpha^2\beta^2} & \frac{-u}{\beta^2} \end{bmatrix}^T \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \equiv \mathbf{F} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \end{aligned} $$

In order to be able to add these twists, they have to be expressed with respect

to a common frame.

$$
\begin{aligned}
{}_{o1}\mathbf{t}_{o1}^{o2} &= \mathbf{S}_{o1}^{f2}\ {}_{f2}\mathbf{t}_{f2}^{o2} + \mathbf{S}_{o1}^{f1}\ {}_{f1}\mathbf{t}_{f1}^{f2} + \mathbf{S}_{o1}^{o1}\ {}_{o1}\mathbf{t}_{o1}^{f1} \\
&= \mathbf{0}_{6\times 1} + \begin{bmatrix} \mathbf{R}_{o1}^{f1} & \mathbf{R}_{o1}^{f1}[\mathbf{p}_{o1}^{f1}]_\times \\ \mathbf{0} & \mathbf{R}_{o1}^{f1} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{2\times 4} \\ \mathbf{I}_{4\times 4} \end{bmatrix} \begin{bmatrix} \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} + \mathbf{F} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{F} & \mathbf{0}_{6\times 4} \\ \mathbf{0}_{6\times 2} & \begin{bmatrix} (\mathbf{R}_{o1}^{f1})_3 & \mathbf{R}_{o1}^{f1}[\mathbf{p}_{o1}^{f1}]_\times \\ \mathbf{0} & \mathbf{R}_{o1}^{f1} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 & \tau_5 & \tau_6 \end{bmatrix}^T \\
&\equiv \mathbf{J}_F \boldsymbol{\tau}
\end{aligned}
$$

where $(\mathbf{R}_{o1}^{f1})_3$ is the 3rd column of $\mathbf{R}_{o1}^{f1}$. $\mathbf{J}_F$ is only dependent on the parameter f and the two variables u and v. Or, expressed in the world frame:

$$
{}_{w}\mathbf{t}_{o1}^{o2} = \mathbf{S}_{w}^{ee}\mathbf{S}_{ee}^{o1}\mathbf{J}_F\boldsymbol{\tau} \tag{6.4}
$$

where $\mathbf{S}_{ee}^{o1}$ is the transformation for the hand-eye calibration (see section 4) and $\mathbf{S}_{w}^{ee}$ represents the robot position. On the other hand:

$$
\left.\begin{array}{l} {}_{w}\mathbf{t}_{w}^{o1} = \mathbf{S}_{o1}^{ee} \cdot {}_{w}\mathbf{t}_{w}^{ee} \\ {}_{w}\mathbf{t}_{w}^{o2} = \mathbf{t}_u \end{array}\right\} \Rightarrow {}_{w}\mathbf{t}_{o2}^{o1} = {}_{w}\mathbf{t}_{w}^{o1} - {}_{w}\mathbf{t}_{w}^{o2} = \mathbf{S}_{o1}^{ee}\mathbf{J}_R\dot{\mathbf{q}} - \mathbf{t}_u \tag{6.5}
$$

where $\mathbf{t}_u$ is the uncontrolled twist of the scene object with respect to the world frame. Combining equations 6.4 and 6.5 :

$$
\mathbf{S}_{o1}^{ee}\mathbf{J}_R\dot{\mathbf{q}} + \mathbf{S}_{w}^{ee}\mathbf{S}_{ee}^{o1}\mathbf{J}_F\boldsymbol{\tau} - \mathbf{t}_u = \mathbf{0} \tag{6.6}
$$

The control constraints can be specified in the form

$$
\begin{bmatrix} \mathbf{C}_R & \mathbf{C}_F \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{U} \tag{6.7}
$$

with $\mathbf{U}$ the control inputs. Consider for instance the following constraints (using only P controllers to simplify the formula):

$$
\left[\begin{array}{cccccc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} k_{q1}(q_{1,des} - q_{1,meas}) \\ k_z(z_{des} - z_{meas}) \\ k_u(u_{meas}) \\ k_u(v_{meas}) \end{bmatrix}
$$

The first row controls the first joint of the robot to a certain value $q_{1,des}$. The second imposes a certain distance between camera (and thus end effector) and scene object. The 3rd and 4th row control a certain pixel to the centre of the image (for example to minimise the probability of the object leaving the field of view). If these criteria are overconstrained, their relative importance is weighted in a pseudoinverse just like in the first case study.

Robot joint equations – in the form of equation 6.6 – can be written for each of the projected blobs a, b, c, ... one wants to control the robot with. Only u and v vary:

$$\begin{bmatrix} \mathbf{S}_{o1}^{ee}\mathbf{J}_R & \mathbf{S}_w^{o1}\mathbf{J}_F^a & \mathbf{0} & \dots \\ \mathbf{S}_{o1}^{ee}\mathbf{J}_R & \mathbf{0} & \mathbf{S}_w^{o1}\mathbf{J}_F^b & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\tau}^{\boldsymbol{a}} \\ \boldsymbol{\tau}^{\boldsymbol{b}} \\ \vdots \end{bmatrix} = \begin{bmatrix} (\mathbf{t}_u)^a \\ (\mathbf{t}_u)^b \\ \vdots \end{bmatrix} \Leftrightarrow \bar{\mathbf{J}}_R \bar{\mathbf{J}}_F \begin{bmatrix} \dot{\mathbf{q}} \\ \bar{\boldsymbol{\tau}} \end{bmatrix} = \bar{\mathbf{T}}$$

All constraints – in the form of equation 6.7 – can also be compiled in a single matrix equation:

$$\begin{bmatrix} \mathbf{C}_R & \mathbf{C}_F^a & \mathbf{0} & \dots \\ \mathbf{C}_R & \mathbf{0} & \mathbf{C}_F^b & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\tau}^{\boldsymbol{a}} \\ \boldsymbol{\tau}^{\boldsymbol{b}} \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{u}^a \\ \mathbf{u}^b \\ \vdots \end{bmatrix} \Leftrightarrow \bar{\mathbf{C}}_R \bar{\mathbf{C}}_F \begin{bmatrix} \dot{\mathbf{q}} \\ \bar{\boldsymbol{\tau}} \end{bmatrix} = \bar{\mathbf{U}} \tag{6.8}$$

$\bar{\mathbf{J}_F}$ is of full rank, since $\mathbf{J}_F^i$ are full rank bases $\Rightarrow \bar{\boldsymbol{\tau}} = \bar{\mathbf{J}}_F^{-1}(\bar{\mathbf{T}} - \bar{\mathbf{J}}_R\dot{\mathbf{q}})$. Combined with equation 6.8: $\bar{\mathbf{C}}_R\dot{\mathbf{q}} + \bar{\mathbf{C}}_F\bar{\mathbf{J}_F}^{-1}(\bar{\mathbf{T}} - \bar{\mathbf{J}}_R\dot{\mathbf{q}}) = \bar{\mathbf{U}}$.

$$[\bar{\mathbf{C}}_R - \bar{\mathbf{C}}_F\bar{\mathbf{J}}_F^{-1}\bar{\mathbf{J}}_R]\ \dot{\mathbf{q}} = \bar{\mathbf{U}} - \bar{\mathbf{C}}_F\bar{\mathbf{J}}_F^{-1}\bar{\mathbf{T}}$$

According to the rank of the matrix $\bar{\mathbf{C}_R} - \bar{\mathbf{C}_F}\bar{\mathbf{J}_F}^{-1}\bar{\mathbf{J}_R}$, this system of equations can be exactly determined, over- or underdetermined. Each of these cases required a different approach to solve it for $\dot{\mathbf{q}}$. These are described in [Rutgeerts, 2007]. More details about inverse kinematics can be found in [Doty et al., 1993]. This section presented two applications of constraint based task specification: [De Schutter et al., 2007] and [Rutgeerts, 2007] contain a more general and in-depth discussion on this approach.

6.3 Visual control using application specific models

This thesis focuses on a general applicable way to estimate the depth of the points that are useful to control a robot arm. If one has more knowledge about the scene, the augmented model will not only simplify the vision, but open up different possibilities to deduct the shape of the objects of interest. Bottom line is not to make a robot task more difficult than it is using all available model knowledge.

6.3.1 Supplementary 3D model knowledge

If the objects in the scene have a certain **known shape**, this extra model knowledge can be exploited. For example, when the projector projects stripes on the

tubes, the camera perceives it as a curved surface since it's looking from a different direction. From this one can deduce the position of the object up to a scaling factor. If the diameter of the tube is also known, one can also estimate the distance between camera (end effector) and the scene. Another example of this technique uses surfaces of revolution as a model. It has been implemented in this thesis in the experiments chapter, section 8.3.

6.3.2 Supplementary 2D model knowledge

If extra knowledge is available about the 2D projection of the objects of interest, extra constraints can be derived from this 2D world. 2D and 3D vision can be combined into a more robust system using Bayesian techniques. Kragic and Christensen [2001] for example propose a system to **integrate 2D cues** with a disparity map, and thus with depth information. Prasad et al. [2006] also studies the possibilities of visual cue integration of 2D and 3D cues, but not using stereo vision as Kragic and Christensen do, but using a time-of-flight camera.
The experiments chapter, section 8.4, contains a fully implemented example of such a case: the model of the object of interest in this case is a cut in living tissue. 2D information like shading or colour contain useful cues that can be used to reinforce or diminish the believe in data from the 3D scanner. Some other interesting 2D cues that are often left unused are:

- **Specular reflection**: Usually, if the scene is known to exhibit specular reflections (smooth surfaces like metallic objects for example), the highlights are looked upon as disturbances. A possible solution to the complication of the specularities, is to identify and remove them mathematically, as in [Gröger et al., 2001]. But they can also be studied as sources of information. If one controls the illumination and knows where the camera is, the Phong reflection model can be used to deduce the **surface normal** at the highlight.

- **Local shape deformations**: Section 3.3 decides that the projected blobs for the structured light technique in this thesis are circle shaped. The section in which these shapes are then segmented (section 5.2) can be refined by also taking into account the **deformation** of these circles. It assumes a locally planar surface, and fit ellipses to the projected circles, to make sure the observed features originated in the projector. However, as one anyhow has the size and orientation of the minor and major axes of each ellipse, it can be used further to **deduct the surface normal** at that point. Augmenting the point cloud with surface normal information, makes triangulation more accurate. This can represent a useful addition, as the point cloud for robotics applications in this thesis is consciously sparse, relative to the point clouds for accurate 3D modelling.
  Note that early structured light methods also used deformation information: Proesmans et al. [1996] for instance makes reconstruction without determining correspondences, only based on the local deformations of a projected grid.

Chapter 7

Software

I quote others only in order to better express myself.

Michel de Montaigne

7.1 Introduction

This chapter discusses software related issues. Section 7.2 explains the software design to make the system as extensible and modular as possible.
Since the processing presented in the previous chapters is computationally demanding, section 7.3 elaborates on what hard- and software techniques can we useful to ensure the computational deadlines are met.

All software presented in this section has been implemented in C++ mainly because the resulting bytecode is efficient: computer vision is computationally demanding. And secondly, the libraries it depends on (*DC1394*, *GLX* and *Orocos*) use the same language, which simplifies the implementation. A downside to C++ however is its large number of programming techniques, that lead to programming errors. Think for example of the confusion among the multiple ways to pass an object: by value, pointer of reference. In more restrictive languages like Java, these problems are automatically avoided. Using Java as an interpreted language on a virtual machine would make the system too slow for these computationally intensive applications. However, a lot of work is being done to make it efficient by compiling Java instead of interpreting it. In addition, standard Java is not realtime: think of the automatic garbage collection for example. *Sun Java Real-Time System* is an effort to make Java deterministic.
Although our implementation is in C++, it does avoid these memory problems by:

- passing objects by reference, and otherwise use the smart pointers (of the STL library): the implementation does not contain normal pointers

- not using arrays, as it opens the gate for writing outside the allocated memory blocks. The appropriate STL library containers are used instead.
- not using C-style casts, as these do not have type checking. Only C++ casts are used: they perform type checking at compile or run time.

7.2 Software design

Figure 7.1 shows a UML class diagram of the OO software design of the system. The boxed classes are subsystems: an I/O wrapper, an image wrapper, the structured light components and the robotics components. The white boxes are components of the presented system, the grey boxes are platform independent external dependencies, and the black boxes are platform dependent external dependencies. Dotted lines represent dependencies. Full lines represent inheritance, with the arrow from the derived classes to the base class.

7.2.1 I/O abstraction layer

The system needs images as input, this is the responsibility of the abstract *inputDevice* class. Any actual input device inherits from this class, and *InputDevice* enforces the implementation of the *readImage* method. Currently two sources are implemented in the system: one for reading files from a hard disk (for simulations), the other for retrieving them from a IIDC-compliant IEEE1394-camera. Other types of camera interfaces can easily be added. The IIDC1394Input class depends on the API that implements the DCAM standard. As the communication with the *FireWire* bus is OS dependent, the box of the DC1394 component is coloured black. The DC1394 library is the Linux way of communicating with the *FireWire* bus. The possibility to interface with the *FireWire* subsystem of another OS can easily be added here. At the time of writing, a second version of the DC1394 library is emerging. Therefore the system detects which of the two APIs is installed, and the *IIDC1394Input* class reroutes the commands automatically to the correct library.
For such detection of installed software, *GNU make* is not sufficient: we chose *CMake* as build system to generate the appropriate *GNU make* scripts. An extra advantage of *CMake* is its platform independence. The classes that inherit from *inputDevice* are parent classes to input classes that specialise in colour or grey scale images. Section 3.3 defends the choice for a grey scale pattern. Hence, the grey scale child classes are the only ones necessary for the SL subsystem. Colour child classes were added to test the performance of the colour implemented pattern separately for the spectral encoding implementation described in section 3.3.2.

The DC1394 library provides **DMA** access to the camera in order to alleviate the processor load: the camera automatically sends images to a ring buffer in the main memory. If one does not retrieve the images from the buffer at a frequency equal to or higher than the frame rate, new image frames are thrown away

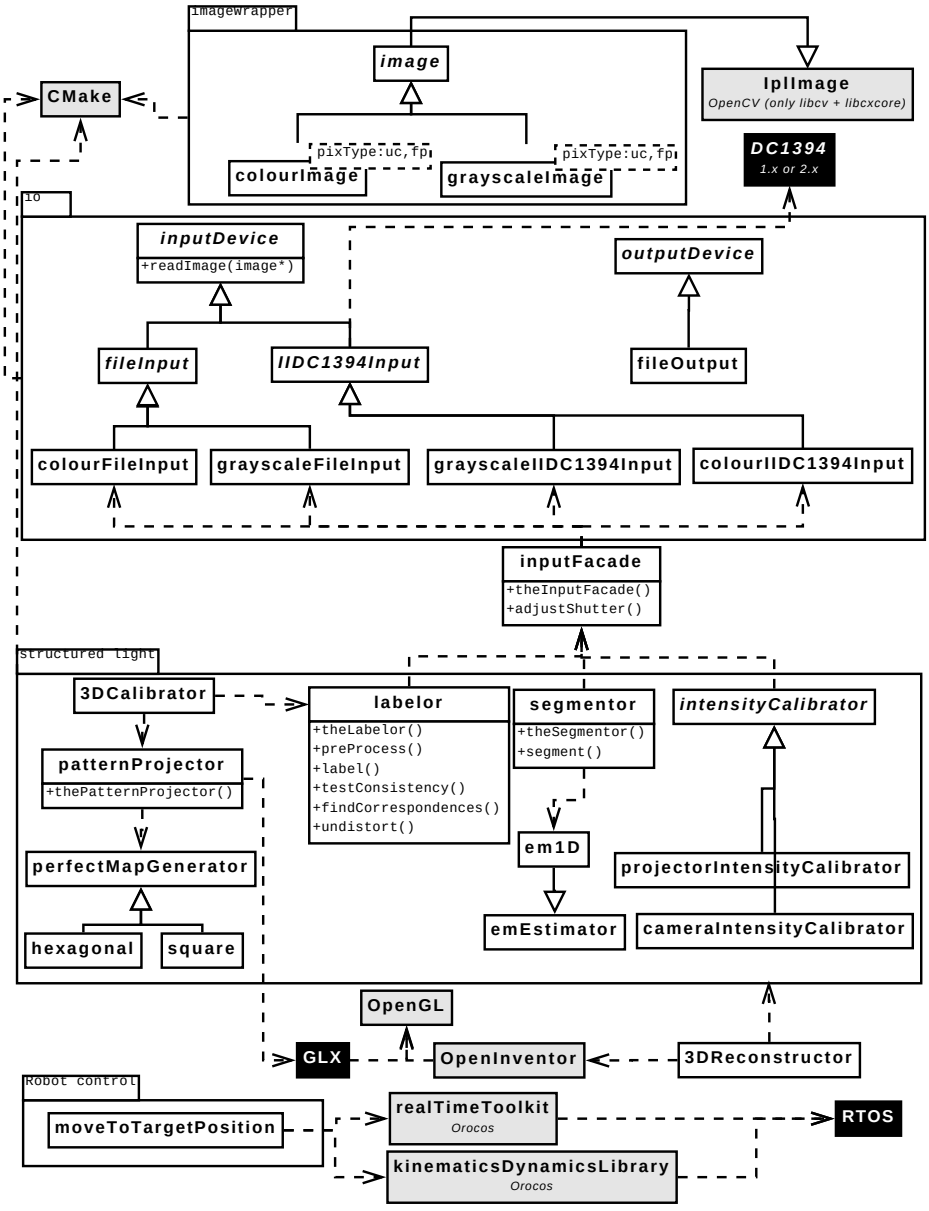


FIGURE 7.1: UML class diagram

as they do not fit into the buffer any longer. For control application, reading obsolete image frames needs to be avoided. A first solution to this problem is to use a separate capturing thread at a higher priority than other threads.

Both the 1.x and 2.x API versions provide a mechanism to flush the DMA buffer. Thus, a second solution is to flush the buffer every time before requesting a new frame. There is a third solution: some cameras support a single shot mode: the constant video stream from camera to main memory is stopped, and the program retrieves a single frame. Our system uses the 3rd possibility if available, otherwise the second.

The *inputFacade* class represents the entire I/O subsystem to the structured light components, thus implementing the **facade** design pattern.
We need only one video input for the structured light subsystem and enforce this here using the **singleton** design pattern, indicated by the *theInputFacade* method in figure 7.1. The traditional way of implementing *singleton* is using **lazy instantiation**. However, this method is not thread safe, and the application is implemented in a multithreaded manner, to be able to work with priorities: visualisation for example needs to have a lower priority than reconstruction, which needs in turn a lower priority than the robot control.
It can be made thread safe again by using mutexes for the lazy instantiation, although **eager instantiation** is a much simpler way to solve the problem. Therefore all classes in the system that implement the *singleton* design pattern use eager instantiation in this system. A downside of eager instantiation is that we do not control the order in which the constructors are executed. Since the classes *segmentor* and *labelor* depend on *inputFacade*, their constructors need to be called after the *inputFacade* constructor. We solve this problem here by moving the critical functionality from the constructors involved to *init* methods in classes *patternProjector*, *labelor*, *segmentor* and *inputFacade*. These *init* methods then have to be called explicitly, enabling the user to define order in which they are executed.

inputFacade also has a method *adjustShutter*, symbolising that the system can semi-automatically adapt the overall intensity of the video stream. First, for cameras that have manual mechanical aperture wheel, the user is given the opportunity to adjust the brightness to visually reasonable levels (the video stream is visualised). Then, using the DCAM interface, the software adjusts the camera settings iteratively such that the brightest pixels are just under oversaturation. First the available *physical* parameters are adapted: integration time t (shutter speed) and aperture, as both influence the exposure:

$$\text{exposure} = log_2 \frac{(f\text{-}number)^2}{t}$$

Especially for cheaper cameras, these can be insufficient, and then we also need to adapt the parameters that *mathematically* modify the output: the brightness (black level offset), gamma γ and gain (contrast) corrections. A simplified formula:

$$\text{output} = \text{input}^{\gamma} \cdot \text{gain} + \text{brightness}$$

7.2.2 Image wrapper

The system uses a I/O abstraction layer to make the structured light subsystem independent of the source of the images. Similarly, we use a wrapper around the image library to make the system independent of the image library. If one wants to use the image processing functions of a different library, only the wrapper (interface) has to be adapted, the rest of the system remains invariant. We chose the OpenCV library, which uses an image datatype called *IplImage*: the only part of the system that uses this datatype is the image wrapper, all other parts use objects of class *image*. *image* is an abstraction that is implemented by two templatised classes representing coloured or grey scale images. The templatisation allows the user to choose the number of colour or intensity levels: currently the wrapper implements 1 or 3 bytes per pixel.

7.2.3 Structured light subsystem

This subsystem is depicted on the lower half of figure 7.1. The *perfectMapGenerator* class implements the algorithms of the section about the pattern logic, section 3.2, of which the details are in appendix A. These have to be executed only once (offline).
The *patternProjector* class controls the projector, and implements the results of *perfectMapGenerator* according to the choices we made in the section about the pattern implementation (section 3.3). As this system uses only one projector, the *patternProjector* class is also singleton.
The projector is attached to the second head of the graphics card, or the DVI/VGA output connector of a laptop, and both screens are controlled independently, to assure that the GUI (and not the pattern) can be displayed on the first screen.
The implementation of the *patternProjector* class has a dependency on an implementation of the OpenGL standard: then the graphics for the projector can be run on the GPU (hardware acceleration), and no extra CPU load is required. The OpenGL features are platform independent, except for the bindings with the window manager. Initially, this thesis used the freeGLUT [1] implementation, but some of its essential features, like fullscreen mode, are not only OS dependent but also window manager dependent. SDL [2] is a cross-platform solution but does unfortunately not allow OpenGL to be active in a second screen while mouse and keyboard are active in the first, at least at the time of writing. Therefore, we had to resort to the combination of a OS dependent and OS independent solution: GLX, the OpenGL extension to the X Window system, is the OS dependent part: a window system binding to Unix based operating systems. The advantage of this project over the *freeGLUT* project is that it implements the *Extended Window Manager Hints* [3]: a wrapper around some of the functionality of X window managers, to provide window manager independence.

[1] http://freeglut.sourceforge.net
[2] Simple Directmedia Layer: http://www.libsdl.org
[3] http://standards.freedesktop.org/wm-spec/latest

The *segmentor* class implements the functionality discussed in section 5.2. *em1D* is a 1D specialisation of *emEstimator*: it clusters the intensity values using a multimodal Gaussian distribution.
The *labelor* class implements the functionality of section 5.3. Both *segmentor* and *labelor* need only one instance, and are thus implemented as singletons. The *preProcess* method is run once offline and constructs a LUT of submatrix codes using the sorting methods of the C++ STL library. *label* finds the 8 neighbours for each blob, *testConsistency* tests the consistency of the grid, *findCorrespondences* uses the LUT to figure out which blobs in the camera image correspond to which blobs in the projector image, if necessary correcting one erroneously decoded blob. *undistort* corrects for radial distortion on the resulting correspondences.
intensityCalibrator is an abstract class to calibrate the relation between incoming or outgoing intensity of an imaging device and its sensory values. As it depends on a visual input, a dependency arrow is drawn from *intensityCalibrator* to *inputFacade*. The two specialisations, for camera and projector, implement the functionality of section 4.2. This calibration can be done offline, or once at the beginning of the online phase. As long as the (soft- or hardware) settings of the imaging devices are not changed, the parameters of the intensity calibration remain the same. Therefore, they are written to a file that is parsed whenever the parameters are needed at the beginning of an online phase.

3DCalibrator uses the correspondences found by *labelor* to estimate the relative 6D orientation between camera and projector, as explained in section 4.4. *3DReconstructor* reconstructs 3D points according to section 5.4.1. The visualisation of the resulting point cloud is through the automatic generation of *Open Inventor* code. *Open Inventor* provides a scripting language to describe 3D primitives in ASCII (or binary) iv-files, it is a layer above any *OpenGL* interface.

7.2.4 Robotics components

Real-time issues

For the control of the robotic arm we use the *Orocos* software [Bruyninckx, 2001], a C++ library for machine control. We use two of its sublibraries: KDL and RTT. The **K**inematics and **D**ynamics **L**ibrary, is the module that is responsible for all mechanical functionality. The **R**eal **T**ime **T**oolkit, is the component that enforces deterministic behaviour. In order to achieve this hard realtime behaviour, it needs to run on a realtime OS. For this, RTAI is used, adding a layer under the Linux OS that works deterministically by only running Linux tasks when there are no realtime tasks to run. Xenomai is another possibility with the same philosophy but does not support *Comedi* at the time of writing, the library for interfacing D/A PCI cards. Otherwise RTLinux is a relevant option, entirely replacing the Linux kernel by a real-time one.

When is real-time necessary? The *io*, *imagewrapper* and *structured light* modules are currently implemented as modules separate from the *Orocos RTT*.

They use OpenGL timers where the process running OpenGL is given a high priority on a **preemptible** Linux kernel: this produces near real time behaviour. The maximal deviations from real time behaviour is such that they are negligible for the vision based task studied in this thesis (with vision control running at order of magnitude $10Hz$ and lower level robot joint control at $\approx 1kHz$).
A *FireWire* bus has two modes: an isochronous and a asynchronous one. Isochronous transfers are broadcast in a one-to-one or one-to-many fashion, the transfer rate is guaranteed, and hence no error correction is available. By design, up to 80 percent of the bus bandwidth can be used for isochronous transfers, the rest is reserved for asynchronous transfers. For IEEE1394a this is maximally $32\text{MB}/s$ for isochronous and minimally $8\text{MB}/s$ for asynchronous data. As the total bandwidth is $49.152\text{MB}/s$, about $9\text{MB}/s$ are "lost" on headers and other overhead ($\pm 40\text{MB}/s$ is usable bandwidth). Asynchronous transfers are acknowledged and responded to, unlike isochronous transfers. In this case the data is time-critical, so the system uses the isochronous mode, in combination with the preemptible Linux kernel.

It is future work to base the system on the *RTT*, such that all components are based on a fully real-time OS layer, to be able to incorporate other sensors at possibly higher frequencies than is acceptable without a real-time OS, with only a preemptible kernel. However, then also the *FireWire* driver needs to work in real-time: the isochronous mode is not deterministic, as it has drift on the receiving of packets depending of the load of interrupts and system calls that the system has to deal with at that time, see [Zhang et al., 2005]. Therefore *RT-FireWire*[4] was developed: currently it is the only project that provides a real-time driver. Via a module emulating an *Ethernet* interface over *FireWire* hardware, *RT-FireWire* enables *RTnet*: hard real-time communication over *Ethernet*, see [Zhang et al., 2005]. It is based on *Xenomai*, a fork of the *RTAI* project formerly known as *Fusion*. However, at the time of writing, *Xenomai* did not provide support for *Comedi*, the Linux control and measurement device interface necessary for robot control through D/A interface cards. Therefore *Orocos* has not been ported to *Xenomai* yet. Fortunately, De Boer [2007] recently ported *RT-FireWire* to RTAI, so that all systems can work under RTAI.
The vision library that is currently used in the project, *OpenCV*, appears to be real-time capable: it does at least not allocate any new memory during capturing or simple vision processing, a full investigation of its real-time functioning is future work.

The experiments by Zhang et al. [2005] conclude that even under heavy interrupt and system call load the maximal drift that arises in the receiving timestamps of the isochronous packets, is $\pm 1ms$. This is an example for *FireWire*, but also more generally as a rule of thumb, for control at frequencies lower than $1kHz$, a preemptive OS with priority scheduling can be a feasible control solution. Section 7.3 will determine that the time frame in which all vision processing can be done on current hardware is in the order of magnitude 10^2ms (the robot control runs at higher frequencies and interpolates between

[4] http://www.rtfirewire.org

vision results). Therefore, real-time vision is not necessary when only this vision sensor is used in the system, isochronous *FireWire* transmission on a preemptible kernel suffices. However, in combination with faster sensors, it may be useful to combine the entire control system fully real-time.

Task sequencing

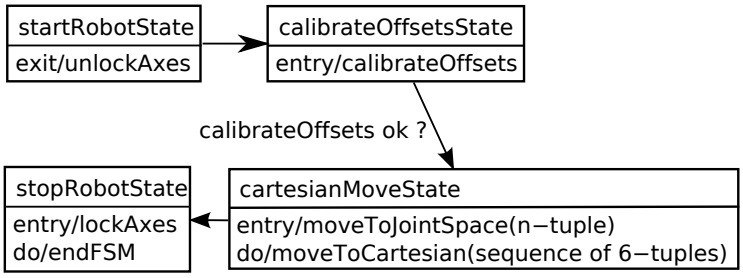


Figure 7.2: FSM

A **finite state machine** is used to describe the sequence of events that execute the mechanical action. Figure 7.2 is a simple example of such FSM. In the *startRobot* state, the brake of the engines are switched off and the engines are powered to keep the joints in position. After a short offset calibration phase, the FSM moves to a motion state. In the *cartesianMove* state, the robot first moves a safe position in joint space, away from singularities. The argument of this function is an n-tuple for a nDOF robot. In our experiments (see chapter 8), $n = 6$. Then the FSM sends the command to execute a sequence of 6D Cartesian coordinates. This sequence is calculated by a simple linearly interpolating path planner. After the motion is complete, the FSM moves to a *stopRobot* state, where the engine brakes are enabled again, the engines are disabled, and the FSM is stopped.
In figure 7.1 *moveToTargetPosition* implements the joint control of the robot arm according to the section about motion control (section 6.2). The twists $\mathbf{t}_u$ used as input in that section, are finite differences coming from the *3DReconstructor* module.

Orocos allows to change control properties without the time-consuming need to recompile [Soetens, 2006], it features:

- *XML* files for the parameters of the controller
- A simple scripting language to describe finite state machines.

7.3 Hard- and software to achieve computational deadlines

7.3.1 Control frequency

To perform the tasks described in the experiments chapter (chapter 8), the robot needs a 3D sensor, but the sensor does not need to have a high resolution. As the robot moves closer to its target, the physical distances between the measured points become smaller. The level of detail in the point cloud increases accordingly, and becomes more local. Hence, the continuous production of point clouds with order of magnitude 10^3 depth measurements is sufficient.

More important than the 3D resolution, is the frequency at which the range data is acquired. Higher frequencies obviously improve the response time. If we can identify an upper limit to the time needed to produce a single point cloud, its inverse is a safe real-time frequency. This section studies a worst case scenario. The time lag between the request for a point cloud and the available point cloud itself consists of:

- Rendering the projector image. The corresponding CPU load is negligible, as the rendering itself is performed by the GPU (through *OpenGL*), but the CPU needs to wait for the result. However, as the GPU is dedicated to this task and only one frame needs to be provided, the overall delay can be neglected. Assuming an LCD projector, when the result is sent to the projector, the LCD needs about 4 to 8 ms to adapt its output to the input.
- Worst case scenario, the available projector image arrives just after the previous update of the projector screen. Assume a refresh frequency of $60Hz$, then the contribution of this delay is $\frac{1}{60}s$.
- The delay caused by the light travelling between projector and camera is of course negligible.
- Worst case scenario, the light arrives in the camera shortly after the previous camera trigger signal. The imaging sensor then has less than a full integration period to integrate the available light, and the resulting image is less bright than it should be and slightly mixed with the previous image. Assuming a $30Hz$ camera, this causes a delay just under $\frac{1}{30}s$. Then we need another $\frac{1}{30}s$ to integrate the pixels of the imaging sensor before transmission can begin.
- As discussed before, IEEE1394a transfers data at 32 $\frac{\text{MB}}{s}$ in isochronous mode. As the DCAM (IIDC) standard specifies uncompressed transmission, the bandwidth is composed of:

$$\Delta f = n_{cam} \cdot \mathcal{H}_c \cdot \mathcal{W}_c \cdot f_r \cdot d$$

with n_{cam} the number of cameras, f_r the frame rate, d the pixel depth (in bytes per pixel).

Assume a camera with a Bayer filter, used by most DCAM compliant *FireWire* cameras, then each pixel is discretised at $1\mathrm{B}/pix$. Thus, supposing the camera uses VGA resolution, transmitting a frame lasts

$$\frac{1}{\Delta f} = \frac{1}{32 \cdot 1024^2 \frac{s}{\mathrm{B}}} \cdot 1\frac{\mathrm{B}}{pix} \cdot 640 \cdot 480 pix \approx 9,2ms$$

- The processing time for image segmentation and labelling as described in sections 5.2 and 5.3.

This is in total 100 ms worst case delay without processing time, supposing the camera is not capable of acquiring and transmitting images in parallel (some are). If the upper bound on the processing time can be limited to $233ms$ (dependent on the CPU), $3Hz$ would be a safe real-time frequency.

7.3.2 Accelerating calculations

In hardware

The better part of the time needed to process a frame is due to calculations. This section describes some of the possibilities to accelerate these calculations. A possibility is to parallelise processing using a second processor in cooperation with the CPU, several options exist:

- using a **smart camera**: a processor integrated in the camera frame processes the video stream instead of the control PC. As vision is about data filtering, the resulting data stream is much smaller than when transmitting the raw images. This is for example advantageous when one cannot use cables to link camera and control PC: a wireless link has a much smaller bandwidth. However, the software on a smart camera is usually dependent on one manufacturer software library, and hence less flexible than a PC system. Initiatives like the CMUcam [Rowe et al., 2007] recently reacted to this situation, presenting a camera with embedded vision processing that is open source and programmable in C.
  If a given smart camera can implement all functionality as described in section 5.2, the segmentation can run on the camera processor while the preprocessing, labelling, triangulation and data management tasks run on the PC. As the computational load of segmentation and labelling are comparable, this is a good work allocation to start with. Responsibilities can be shifted between camera processor and CPU depending on the concrete hardware setup, resulting in a roughly doubled vision processing frequency as two processors are working in parallel. If the processor in the camera can process this information in parallel with acquiring new images, the processing time is again shortened by at least two camera frame periods (e.g. by $\frac{2}{30}s$), as explained in section 7.3.1.

- **g**eneral-**p**urpose computation on **g**raphical **p**rocessing **u**nits (GPGPU) has expanded the possibilities of GPUs recently. Before GPUs provided a limited set of graphical instructions, now a broad range of signal processing functionalities is reachable using the C language. [Owens et al., 2007] Recently both competitors on this market released their GPU API: *ATI/AMD* provides the *Close to metal* interface, and *Nvidea* call theirs *Cuda*.

- Other processors with parallel capabilities are worth considering: **p**hysics **p**rocessing **u**nits (PPU), or **d**igital **s**ignal **p**rocessors (DSP). All architectures have their advantages and disadvantages, it is beyond the scope of this thesis to discuss them.

- The use of a **f**ield-**p**rogrammable **g**ate **a**rray (FPGA) is another interesting possibility, using highly parallelisable reprogrammable hardware.

- Using a second CPU on the same PC, or on a different PC:
  A *FireWire* hub is another way to achieve parallel computation. We successfully tested a IEEE1394a hub to split the video stream to 2 (or more) PCs. If the processing power is insufficient, each of the PCs can execute part of the job (segmentation, labelling ...). Visualising the video stream on the computer screen is a considerable part of the processing time. Therefore, the simplest possibility to alleviate some of the work of the control PC is to use the setup as depicted in figure 6.1: the PCs do not need to communicate. A more balanced and more complex solution is to run the vision work on one PC, and visualisation plus robot and projector control on another. Then the results of the vision need to be transferred to the control PC, for example over Ethernet, see figure 7.3. If the data needs to be delivered real-time, RTnet – a deterministic network protocol stack – is a good solution. Otherwise, solutions in the application layer of a standard protocol stack, such as the Real Time Streaming Protocol will do: Koninckx [2005] describes such a system. Contrary to what its name leads to believe, this protocol does not provide mechanisms to ensure timely delivery. See section 7.2.4 to decide whether real-time behaviour is required.
  Since the transmitted data is a stream of sparse point clouds, the data does not require much bandwidth (as opposed to the situation where the video stream would have to be transmitted).

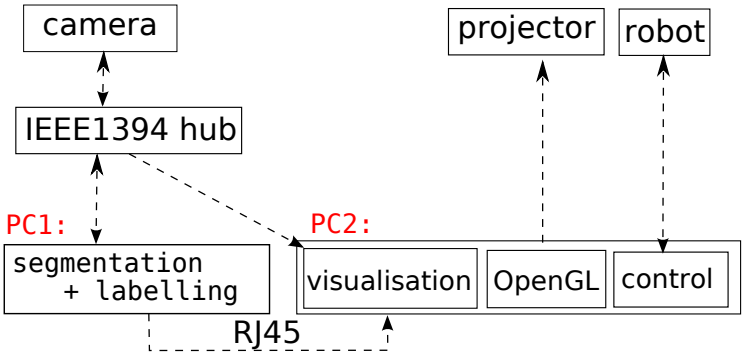


Figure 7.3: Accelerating calculations using a hub

In software

Efficiency is not only a matter of efficient hardware, but also of efficient software. We **avoid superfluous calculations** using:

- Tracking (local search) instead of repeated initialisation of the projected features. (global search)
- Image pyramids: processing parts of the image at a lower resolution when detailed pixel pro pixel calculations are not necessary, by subsampling.
- A further selection of parts of the image that need processing, based on the task at hand, avoids unnecessary calculations.

Flexible software It is a design aim to make the system portable to more, or less powerful systems. On faster systems the software then produces data at a higher frequency, on a slower system it **degrades gracefully**. In other words, if point cloud calculations tend to last too long in comparison to a required minimum control frequency for the robot, the spatial resolution should degrade gradually as less computing power is available. The easiest way to do this is to adapt the number of features in the projector image to the available computing power. The different threads should run at different **priorities** to ensure this graceful degradation.

7.4 Conclusion

The first part of this chapter presented a modular software architecture, minimising the effect of changing one component on the others. These components include an I/O abstraction layer, an image format abstraction layer, a structured light subsystem and a robot control subsystem.
As this is a computationally demanding application, the second part of the chapter describes the options available to adapt the system to satisfy these needs. If software optimisations are inadequate, the help of one of several kinds of coprocessors can be useful.

Chapter 8

Experiments

La sapienza è figliola dell'esperienza.

Leonardo da Vinci

8.1 Introduction

This chapter explains the robot experiments. First in section 8.2, a general object manipulation task using the sparse pattern described in section 3.3.6. Then an industrial one, deburring of axisymmetric objects, presented in section 8.3. The last part, section 8.4, describes a surgical application: the automation of a suturing tool. All experiments were done using a **6DOF robot arm**, in this case a Kuka-361. Another element that these three experiments have in common, is that they go beyond 3D reconstruction, but need to interprete the data to be able to perform the robot task. They need to **localise** the features of interest in the scene.

The reader will notice that the last two experiments mainly use a different type of structured light than the 2D spatial neighbourhood pattern described throughout this thesis. However, as stated in these experiments, that pattern is equally useful in those experiments: experimentation with other patterns was mainly done to be able to compare with existing techniques.

8.2 Object manipulation

8.2.1 Introduction

This experiment applies the sparse 2D spatial neighbourhood structured light technique explained in section 3.3.6 on arbitrary objects, in this case a approximately cylinder shaped surface. Possible applications of this technique are:

- automatic **sorting** of parcels for a courier service
- automatic **dent removal** on a vehicle's body
- automatic **cleaning** of buildings, vehicles, ships ...
- automatic **painting** of industrial parts in limited series. The same applies to mass production, but in that case it is probably more economical to arrange for a known structured environment, using a preprogrammed blind robot. Starting from the depth information from the structured light, a path for the spray gun can be calculated. Vincze et al. [2002] present such system. They use a laser plane to reconstruct the scene, a time multiplexed technique: they thus need a static scene (and end effector if the camera is attached to it) while scanning the object. This can be improved using our single-shot structured light technique. The reconstruction will be more sparse, it sufficient to execute this application. The detailed 3D reconstruction a laser plane produces, is not needed in every part of the object. Some parts may need to be known in more detail, and some more coarsely. Then a stratification of 3D resolution similar to the ones proposed in sections 8.3.1 and 8.4.4 eliminates superfluous computing time.

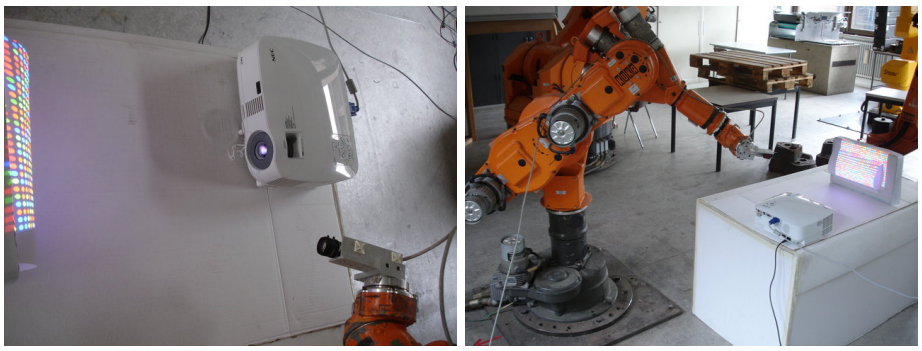


Figure 8.1: Experiment setup

8.2.2 Structured light depth estimation

Encoding

Pattern logic and implementation We implement this procedure with the pattern implementation of figure 3.9, but using a larger – less constrained – perfect map: with $h = w = 3$, $a = 6$ (see figure 3.7 on the left). $a = 5$ results is a sufficient resolution, but is prime. In that case, one is constrained to using only one type of pattern implementation as a prime number cannot be factorised in numbers larger than one. A **combination of several cues** is often more interesting (see section 3.3) since the implementation types are orthogonal: the one does not influence the other. Hence, the total number of possibilities is the **multiplication** of the possibilities in each of the pattern implementation domains. Thus, in each of the domains, only a very small number of different elements is needed. And the smaller the number of elements (the coarser the discretisation), the more robust its segmentation.

In this case, we choose to use 3 colours and 2 intensities. Using spectral encoding limits the application to near white surfaces, or imposes the need to adapt the projected colours to the scene, as section 3.3.2 describes. This implementation can easily be altered to the implementation chosen in 3.3.6, that does allow for coloured scenes without having to estimate the scene colours.

Pattern adaptation The sizes of the projected blobs are adapted to the size that is most suitable from the camera point of view for each position of the robot end effector: not too small which would make them not robustly detectable, and not to large which would compromise the accuracy (see section 3.4).

Decoding

Segmentation The camera image is converted to the **HSV** colour space, as that is a colour space that separates brightness and frequency relatively well (see section 3.3.2). As the intensity of the ambient light is negligible compared to the intensity of the projector light, blob detection can be applied to the V-channel. The median of the H and V-values of each blob is used to segment the image.

Labelling Since invariance to only 4 rotations is imposed (see section 3.2.5), one needs to find the orientation of the two orthogonal directions to the 4 neighbours of each blob that are closest, and the orientation of the two orthogonal directions to the 4 neighbours that are furthest away (the diagonals). This is done blob by blob. Then an **EM** estimator uses the histogram of these orientations to automatically determine the angle which will best **separate** the **nearest and** the **furthest neighbours**. This angle is then used to construct the graph that connects each blob to both kinds of neighbours. First the closest four blobs in the expected direction are localised. Then the location of the diagonal neighbours is predicted, using the vectors between the central and the closest four blobs. The location of the diagonal neighbours is then corrected

choosing the blobs that are closest to the prediction.
Now the **consistency of the connectivity graph** is checked. Consider a rotated graph such that one of the closest neighbour orientations points upwards (the algorithm does not need to actually rotate the graph: this is only to define left, right, up and down here). Then, the upper neighbour needs to be the left neighbour of the upper right neighbour, and at the same time the right neighbour of the upper left neighbour. Similar rules are applied for all neighbouring blobs. All codes that do not comply to these rules are rejected.

Now each valid string of letters of the alphabet can be decoded. Since this will be an online process, efficiency is important. Therefore, the base 6 strings of all possible codes in all 4 orientations are quicksorted in a offline preprocessing step. During online processing, the decoded codes can be found using binary search of the look up table: this limits the complexity to $O(\log n)$.
We use **voting** to correct a single error. If a code is found in the LUT, then the score for all elements of the code in the observed colour is increased by 9: once for each hypothesis that one of the neighbouring blobs and the central blob is erroneous. If the code is not found, then for each of the 9 elements we try to find a valid code by changing only one of the elements to any of the remaining letters in the alphabet. If such code can be found, we increase the score for that colour code of that blob by one. Then we assign the colour to the blob that has the highest vote, and decode the involved blobs again.

Calibrations The **geometric calibration** of camera and projector was done using a calibration grid and binary Gray code patterns both in horizontal and vertical direction [Inokuchi et al., 1984]. Section 4.4.3 explains this technique, the Gray code maximises the Hamming distances between consecutive code words. Because of the lower robustness of the associated calibration algorithms, and the need for online updating of extrinsic parameters, the implementation of the self-calibration procedure of section 4.4.4 would improve the experiment.
The only deviation from the pinhole model that is compensated for in camera and projector, is the one that has most influence on the reconstruction: **radial distortion**. We use the technique by Perš and Kovačič [2002] to that end, see section 4.3.3. The asymmetry of the projector projection is accounted for using the extended (virtual) projection image of section 4.3.2.

The experiment performs a colour calibration in such a way that the distance between the colour and the intensity values of the blobs in the camera image, which correspond to different letters of this alphabet, is as large as possible. This calibration is required as both the camera and the projector have different **non-linear response functions** for each of the colour channels.
The projected colours are not adapted to the colour of the scene at each point here: we assume that the scene is not very colourful. For an arbitrary scene this adaptation would have to be done, see section 3.3.2. No compensation for specular reflections was implemented in this experiment: we assume Lambertian reflectivity.

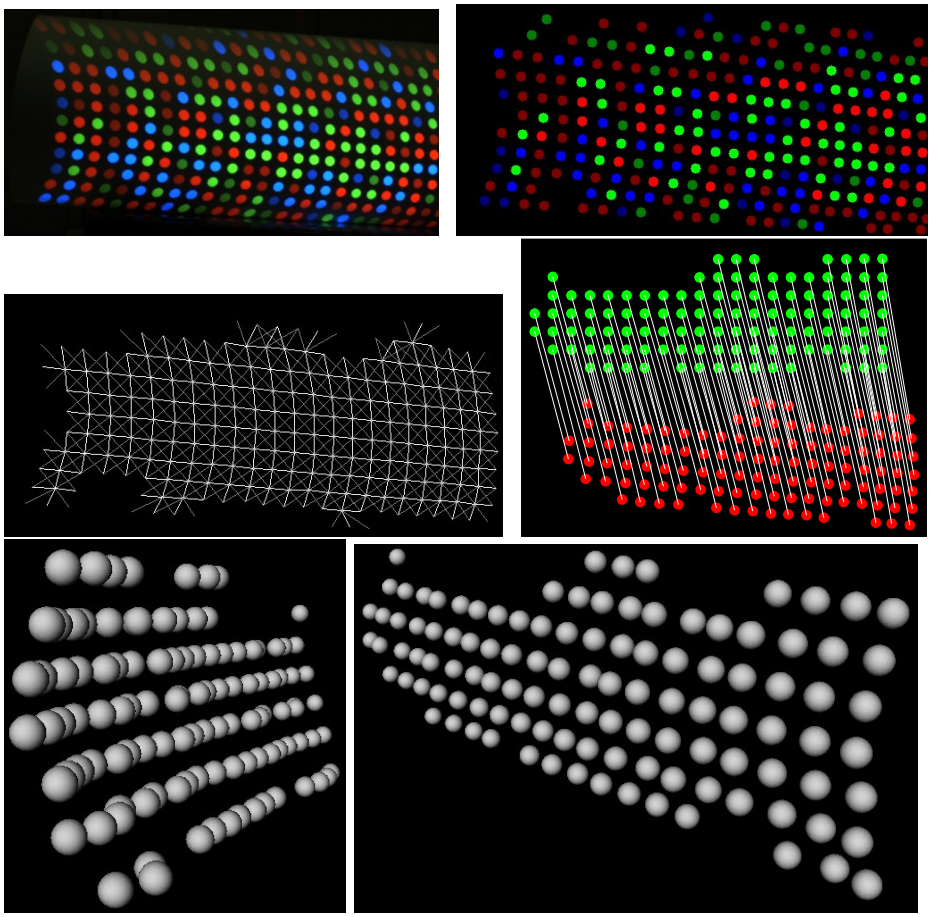


FIGURE 8.2: Top left: camera image, top right: corresponding segmentation, middle left: labelling, middle right: correspondences between camera and projector image, bottom: corresponding reconstruction from two points of view

Reconstruction The correspondences are then used to reconstruct the scene, as shown in figure 8.2: this implies solving a homogeneous system of equations for every point, by solving the eigenvalue problem described in section 5.4.1.

Robot control

Using the 3D point cloud, we calculate a target point for the robot to move to, a few *cm* in front of the object. As the projection matrices are expressed in the camera frame, this point is also expressed in the camera frame. An offline hand-eye calibration is performed in order to express the target point in the end effector frame. Then the encoder values are used to express the point in the robot base frame, as schematically shown in figure 6.2. Our robot control

software [1] calculates intermediate setpoints by interpolation, to reach the destination in a specified number of seconds. At each point in time, all information is extracted from only one image, so online positioning with respect to a moving object is possible. A good starting position has to be chosen with care, in order not to reach a singularity of the robot during the motion.
During online processing, the blobs are tracked over the image frames, for efficiency reasons.

8.2.3 Conclusion

This experiment showed the functionality of the sparse 2D structured light technique presented in this thesis. It estimates the distance to the scene at a few hundred points, such that a robotic arm can be positioned with respect to the scene objects. The pattern can be used online at a few Hz when the extrinsic parameters are adapted with the encoder values.

8.3 Burr detection on surfaces of revolution

8.3.1 Introduction

The use of 3D scanning for the automation of industrial robotic manufacturing is rather limited. This is among other reasons because the reflectivity of objects encountered in these environments is often far from ideal, and the level of automation of the 3D acquisition is still limited. This section describes how to automatically extract the location of geometrical irregularities on a surface of revolution. More specifically, a partial 3D scan of an industrial workpiece is acquired by structured light ranging. The application this section focuses on is a type of quality control in automated manufacturing, in this case the detection and removal of burrs on possibly metallic industrial workpieces.



Figure 8.3: From left to right: a Kuka-361 robotic arm, the test object used and a high resolution range scan of this object with specular reflection compensation.

[1] Orocos [Bruyninckx, 2001]

Specular reflections make cylindrical metallic objects virtually impossible to scan using classical pattern projection techniques. A highlight occurs when the angle between camera and surface normal is equal to the angle between the light source and the surface normal. As the scene contains a surface of revolution, it has a variety of orientations. Hence, at some scene part a highlight due to the projector will almost always be cast into the camera. In order to avoid this infamous over- and under-saturation problem in the image we propose two structured light techniques adapted to specular reflections. The first one adapts the local intensity ranges in the projected patterns, based on a crude estimate of the scene geometry and reflectance characteristics [Claes et al., 2005]. The second one is based on the relative intensity ratios of section 3.3.7, in combination with the pattern adaptation in terms of intensity of section 3.4. Hence, these highlights are compensated for by the projector.

Secondly, based on the resulting scans, the algorithm will automatically locate artefacts (like burrs on a metal wheel) on the surface geometry. The techniques of constraint based task specification (see section 6.2.3) can then be used to visually control a robotic arm based on this detection. The robot arm can operate a manufacturing tool to correct (deburr) the workpiece.
This section focuses on burr detection on a surface of revolution. To do so, the axis of this object and corresponding generatrix should be determined automatically from the data. The generatrix is the curve that generates the surface of revolution when it is turned around the axis of the object. The triangular mesh (or point cloud) produced is used to detect axis and generatrix. Next a comparison of the measured surface topology and the ideal surface of revolution (generated by the generatrix) will allow to identify the burr.

The following paragraph is an overview of the strategy used here, details will become clear in the next sections.
The search space for finding the rotational axis is four dimensional: a valid choice of parameters is two orientation angles (as in spherical coordinates) and the 2D intersection point with the plane spanned by two out of three axis of the local coordinate system. In figure 8.7 these 4 coordinates are the angles θ and ϕ, and the intersection (x_0, y_0). For finding the axis we test the circularity of the planar intersections of the mesh in different directions, using statistical estimation methods to deal with noise. Finally the 'ideal' generatrix derived from the scan data is compared to the real surface topology. The difference will identify the burr. The algorithm is demonstrated on a metal wheel that has burrs on both sides.

Literature overview

This section presents previous work on the reconstruction of the rotational axis and generatrix based on 3D data points. Qian and Huang [2004] use uniform sampling over all spatial directions to find the axis. The direction is chosen where the intersection of the triangular 3D mesh with planes perpendicular to

the current hypothesis for the axis best resembles a circle. This is done by testing the curvature between the intersection points (defined as the difference between the normals divided by the distance between subsequent 2D vertices), which should be constant for a circle. In our method a similar technique is used, but the difference is that the data of Qian and Huang has to be ϵ-densely sampled, meaning that for any x on the reconstructed surface there is always an x_i on the mesh such that $\| x - x_i \| < \epsilon$ for a fixed positive number ϵ. In other words, data points have to be sampled all around the surface of revolution. In our case however, the data is partial: only one side of the object is given.

Pottmann et al. [1998, 2002] give an overview of methods based on motion fields, requiring the solution of a generalised eigenvalue problem, an interesting alternative to the method presented in this section. Orriols et al. [2000] use Bayesian maximum likelihood estimation. In their work a surface of revolution is modelled as a combination of small conical surfaces. An iterative procedure is suggested where the axis parameters are estimated first, then the generatrix is estimated. Using the latter they make a better estimate of the axis, then again the generatrix etc. Their main interest is a precise reconstruction of the surface of revolution, not the application of an online estimation procedure. In our case, the roughest 3D reconstruction that can identify the burr is sufficient, the rest are superfluous calculations. The computational complexity of the algorithm is not discussed in the paper either, but seems to be too high for this application where speed is relevant.

The rest of the experiment is organised as follows: subsection 8.3.2 gives an overview of the structured light scanning. Section 8.3.3 discusses the axis localisation, and subsection 8.3.4 the burr detection. Results are shown in subsection 8.3.5.

8.3.2 Structured light depth estimation

Before explaining how to do axis retrieval and burr detection we give a concise overview of two possible structured light techniques. For both techniques local adaptations of the pattern intensity are needed to compensate for possible highlights.

Temporal encoding

As burrs are relatively subtle deformations, one could use the overall high resolution 3D reconstruction of figure 8.3 on the right. To that end, use a temporal pattern encoding, as discussed in section 3.3.4. More precisely, 1D Gray coded patterns, augmented with phase shifting to improve the accuracy.

Specular reflection (on metallic objects) results in oversaturation in the highlight area, or undersaturation in the other areas if one tries to compensate the highlight by reducing the light intensity. Interreflections and scattering will make that even for binary patterns not only the pixels in the image resulting from fully illuminated parts of the pattern are affected, but the pattern will rather be “washed out” in a complete area. In case the pattern is combined

with interferometry [Scharstein and Szeliski, 2003] artefacts tend to occur even sooner. As interferometry uses shifted 2D sine wave patterns, local over- or undersaturation will make that the reflected sine patterns will be clipped at the saturation level of the camera. A strong periodical artifact occurs, see figure 8.4.

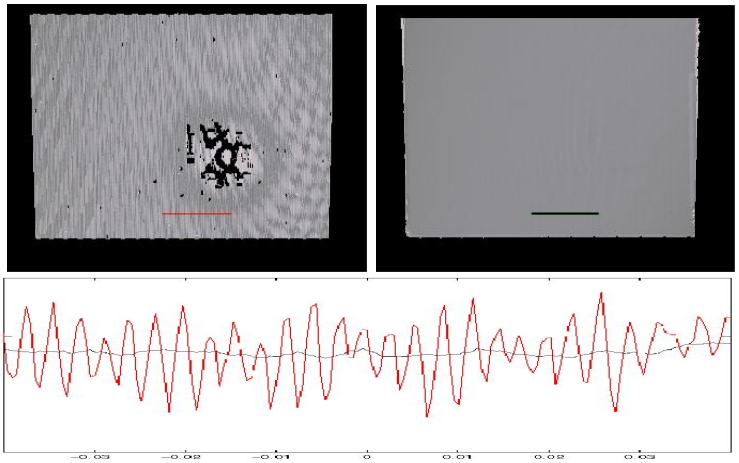


FIGURE 8.4: Top left: incomplete reconstruction due to camera oversaturation. Top right: corrected planar geometry using the technique of Claes et al. [2005]. Bottom: cross section for the line indicated above. One can see the artifact due to level clipping, the circular hole is due to the mirror reflection of the projector.

To overcome this problem a two step approach is used. First, a crude estimation of local surface geometry and reflectance properties are made. For the geometry a low resolution time coded approach is applied. Extended filtering removes data with too much uncertainty. Missing geometry is interpolated using thin plate spline surfaces. The reflectance properties are taken into account on a per pixel basis by modelling the path from camera to projector explicitly. The test patterns needed for this are submerged in the sequence of shots needed for the geometry, for details see [Koninckx et al., 2005].

Next, a per pixel adapted intensity in the projection pattern is used to force the reflected radiant flux originating from the projector within the limited dynamic range of the camera. On the photometric side, nonlinearities and crosstalk between the colour channels are modelled for both the camera and projector. The system can now deal with a wider category of objects. Figure 8.5 illustrates the result on a metal wheel, which will be used as the example throughout the remainder of this section. The quality of the reconstruction improves and the reconstructed area is considerably larger. To test the usefulness of this temporal technique in the context of deburring, we reused software implemented by Koninckx et al. [2005].



Figure 8.5: Left: reconstruction with and without taking surface reflectance into account. Right: uniform sine pattern and plane white illumination as seen from the camera.

Spatial encoding

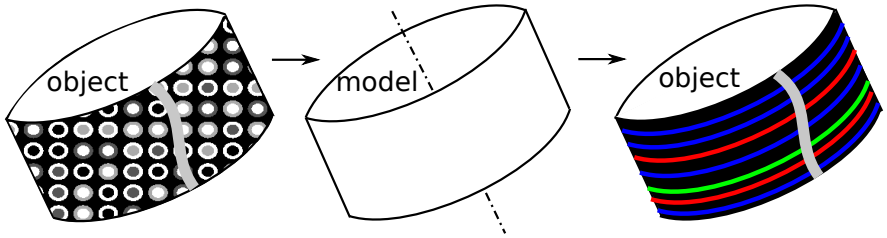


Figure 8.6: A structured light process adapted for burr detection

Burrs require detailed scanning of the surface in certain image parts (where the burrs are located), and very coarse scanning in others (only to detect the axial symmetry). Therefore, ±99% of the reconstruction data of the high resolution scan of the previous paragraph can be discarded, only the data around the burrs is needed in such detail. Hence a lot of computations are superfluous, not an interesting situation for a system where computational load is the bottleneck: it is primordial for our algorithm to be fast. A possible way to avoid this problem, is using the following two step technique (see figure 8.6):

- A **coarse overall depth estimation**. Apply the 2D structured light technique of section 3.3.6 with uniformly distributed features, as one has no knowledge of the position of the object yet, let alone the burrs.
- A **detailed local depth estimation**. Apply the model based visual control techniques of section 6.3: project a 1D pattern with lines, such that the intersection between the projection planes and the axis of symmetry is approximately perpendicular. These 1D spatial neighbourhood patterns

are discussed in section 3.2.3.3. One does not need a dense pattern as the one by Zhang et al. [2002] or Pagès et al. [2005], rather a sparse pattern like the pattern by Salvi et al. [1998], where one of the two line sets, one of the two dimensions, is removed. A De Bruijn sequence is encoded in the lines: observing a line and its neighbouring lines in the camera image, identifies a line in the projector image. As the setup is calibrated, this line corresponds to a known plane in 3D space, that intersects with the just estimated pose of the surface of revolution model. There is no need to calculate the locus of the entire intersection. Detect the point where the deformation of the observed line is least in accordance with what is to be expected from the surface of revolution. Combining the different strongest deformations of the different projected lines, results in the most probable location of the burr.
This technique is an example of **active sensing**: the projected pattern is adapted based on previous observations to retrieve missing information. To make the algorithm more robust, one can keep several hypothesis of likely burr locations for every stripe, and detect the complete burr curve with a condensation filter, see section 8.3.4.

This is a combination of two single shot techniques, and therefore a technique that allows for relatively fast movement of the scene. This is another advantage of this approach, as the 1D Gray code pattern with phase shifting, presented in the previous paragraph, needs tens of image frames to make the reconstruction, and thus requires a static scene, which is rarely the case in a robot environment. However, to test the feasibility of the burr detection algorithm, the remainder of this section uses the 1D Gray code structured light.

8.3.3 Axis reconstruction

After having explained how the mesh is generated, the mesh data will now be analysed. First, the axis of the corresponding surface of revolution is to be estimated. Afterwards section 8.3.4 detects the geometrical anomaly.

Overview

We reconstruct the axis of the surface of revolution corresponding to the triangular mesh. The data points are not sampled uniformly around the axis. Determining the axis is a 4D optimisation problem. A possible choice for the parameters is the angle ϕ with the Z axis, the angle θ with the X axis in the XY plane, and a 2D point (x_0, y_0) in the XY plane.

A well chosen selection of all possible orientations of the axis is tested. Which ones is explained in the next paragraph, first we explain how the testing is done. For each of the orientations, construct a group of parallel planes perpendicular to it. Along the line defined by that orientation there is a line segment where planes perpendicular to it intersect with the mesh. The planes are chosen uniformly along that line segment (see figure 8.7)

Test the circularity of the intersection of those planes with the mesh, let the resulting error be $f(\theta, \phi)$ (as explained in the next subsection with the algorithm details). Retain the orientation corresponding to the intersections that best resemble circles. This orientation is an estimate of the axis orientation and hence determines two out of four parameters of the axis.

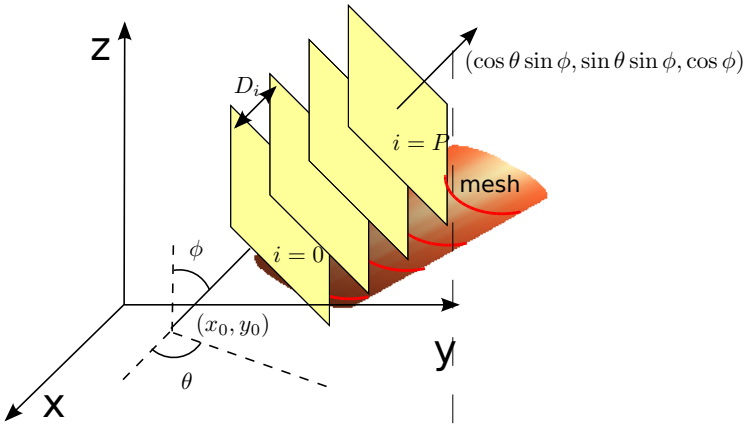


Figure 8.7: Determining the orientation of the axis.

First, consider all possible orientations the axis can have (2D: θ and ϕ). Sample them in a uniform way (see figure 8.8). To sample points uniformly on the surface of a unit sphere it is incorrect to select spherical coordinates θ and ϕ from uniform distributions, since the area element $d\Omega = sin(\phi)d\theta d\phi$ is a function of ϕ, and hence points picked in this way will be closer together near the poles. To obtain points such that any small area on the sphere contains the same number of points, choose

$$\theta_i = \frac{i\pi}{n} \text{ and } \phi_j = cos^{-1}(\frac{2j}{n} - 1) \text{ for } i, j = 0...n-1$$

Test each of the orientations and keep the one with the minimal error: call it $f(\theta^*, \phi^*)$.

Secondly, use steepest descent to diminish the error further: the estimate just obtained has to be regarded as an initialisation in order to avoid descending into local minima. Thus, its sampling is done very sparsely, we choose $n = 3$ (9 evaluations).

Gradients are approximated as forward differences (of the orientation test f) divided by the discretisation step $\Delta\theta$. We choose $\Delta\theta = \Delta\phi = \frac{\pi}{2n}$. Then

$$\begin{bmatrix}\theta_{i+1}\\ \phi_{i+1}\end{bmatrix} = \begin{bmatrix}\theta_i\\ \phi_i\end{bmatrix} - s\nabla f \text{ with } \theta_0 = \theta^*,\ \phi_0 = \phi^*$$

and s the step size, choose $s = \frac{\Delta\theta}{\|\nabla f\|_2}$. If $f(\theta_{i+1}, \phi_{i+1}) \geq f(\theta_i, \phi_i)$ then s was

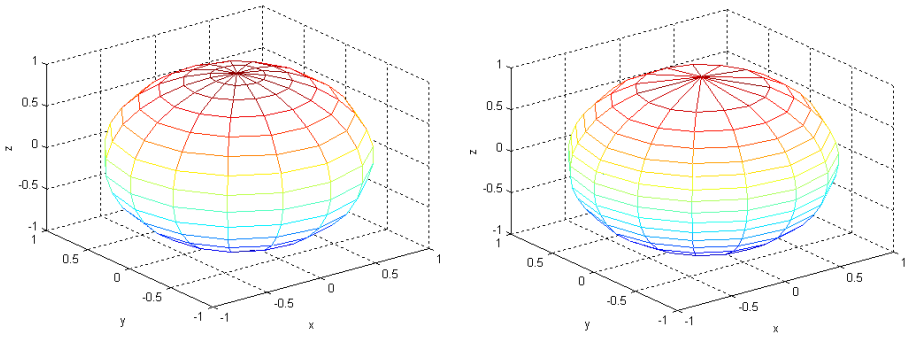


FIGURE 8.8: Left: non uniform point picking $\phi_j = \frac{j\pi}{n}$, right: uniform point picking $\phi_j = cos^{-1}(\frac{2j}{n} - 1)$

chosen too big: $s_{j+1} = \frac{s_j}{2}$ until the corresponding f is smaller. If it does not become smaller after say 5 iterations, stop the gradient descent.
Then do a second run of steepest descent, using a smaller discretisation step: $\Delta\theta = \frac{\pi}{2n^2}$, using the same step size and stop criterion. The result can be seen in fig.8.9.

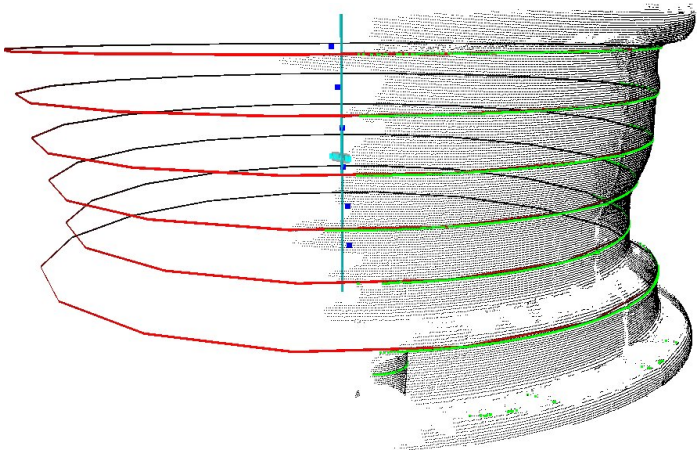


FIGURE 8.9: Axis detection result: estimated circles in red, mesh points in black, intersection points in green, circle centres in blue in the middle.

In this way, determining the axis orientation takes about 30 evaluations of f. Gradient descent descends only slowly into minima that have very different medial axes, therefore using Levenberg-Marquardt might seem a good idea. Unfortunately, each Levenberg-Marquardt step requires computing a Newton Rapson step size, and thus the computation of the Hessian of f. To approx-

imate each Hessian using forward differences five costly evaluations of f are necessary, and even then the step size is to be determined as a line minimum, requiring even more evaluations of f in each step size. In this case, the problem is about symmetrical for θ and ϕ (as can be seen in fig.8.12), therefore gradient descent converges reasonably fast.

The test on circularity returns the centre and radius of the best circle that can be fitted to the data. Therefore the other two parameters of the axis—determining the location in space—can be estimated from those circle centres that come as outputs of the best axis orientation retained. For algorithm details and complexity evalution, see appendix C.1. It concludes that the complexity is O(V). Therefore, reducing V will increase the speed substantially. The mesh discussed contains 123×10^3 triangles. The implementation that uses all triangles currently takes several seconds to complete (at $\approx 1Ghz$), too much for an online version. The better part of that time is spent on the axis detection part. A solution is mesh simplification, but a much more interesting solution is to not calculate such detailed mesh in the first place: use the spatially encoded structured light technique as proposed in section 8.3.2.

As the two angles of orientation of the axis have been found, we now have to determine the two other parameters to fix the axis: its intersection with a plane spanned by two of the three axes of the local coordinate system. All estimated circle centres are located along the axis. Project these noisy 3D circle centres onto a plane perpendicular to the axis orientation. Then determine the most accurate axis centre. Averaging these 2D circle centre points would incorporate outliers. Hence, a different approach is used: a RANSAC scheme eliminates the outliers and determines which point is closest to the other points. Now all four parameters of the axis have been determined.

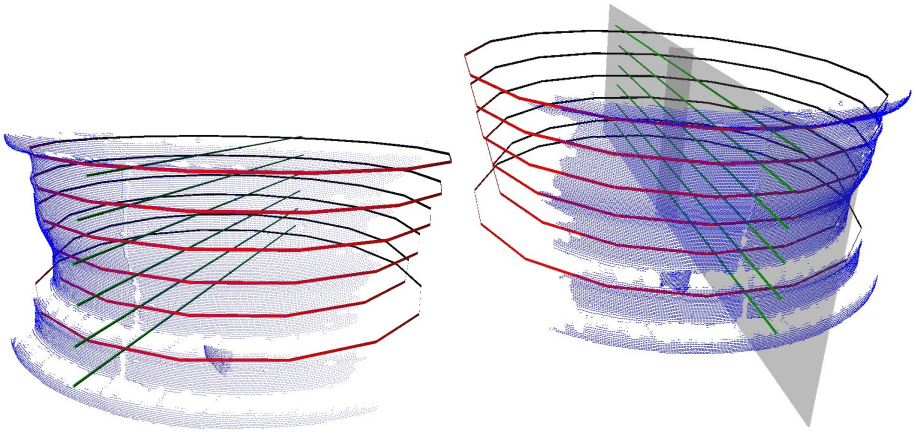


Figure 8.10: Left: rays indicating where the mesh differs most from the estimated circles. Right: a transparent plane indicates the estimated axis and intersects the mesh at the location of the burr.

8.3.4 Burr extraction

In the algorithm described below, a voting algorithm is used (RANSAC) to deal with noise. RANSAC can be looked upon as a Bayesian technique in the sense that it is similar to a particle filter without a probability distribution function. Applied to this case: for every intersection perpendicular to the generatrix, all angles at which the burr could be located except for one are discarded. This one angle, the one with the strongest geometric deviation, called angle α_i (see the right hand side of figure 8.11). To make this part of the algorithm more robust, one could also take multiple hypothesis into account, and work with a particle filter, the price to pay is the extra computational cost. This approach is similar to the condensation filter by Isard and Blake [1998], where he uses equidistant line segments along which multiple hypothesis for the strongest cue may occur. Also here, the intersections are equidistant as no information is available that leads to a data driven discretisation choice (see the left hand side of figure 8.11).

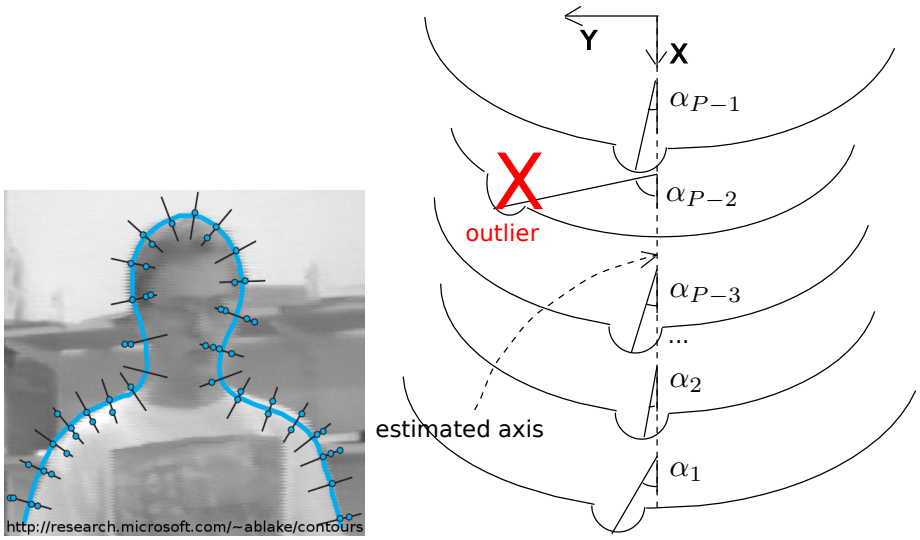


FIGURE 8.11: Left: equidistant line segments as low level cue in the condensation filter by Isard and Blake; right: determining the burr location given the axis.

Given the 4 coordinates of the axis, the surface can then be represented using 2 DOF: the position along the axis and the radius at that position. This is the generatrix (as can be seen on the left of fig.8.14). Now that the axis is estimated, comparing the mesh data to the ideal generatrix leads to the burr angle α, see appendix C.2. We now have determined the axis and the burr angle relative to the axis, uniquely identifying the burr in 3D space.

8.3.5 Experimental results

Axis orientation detection

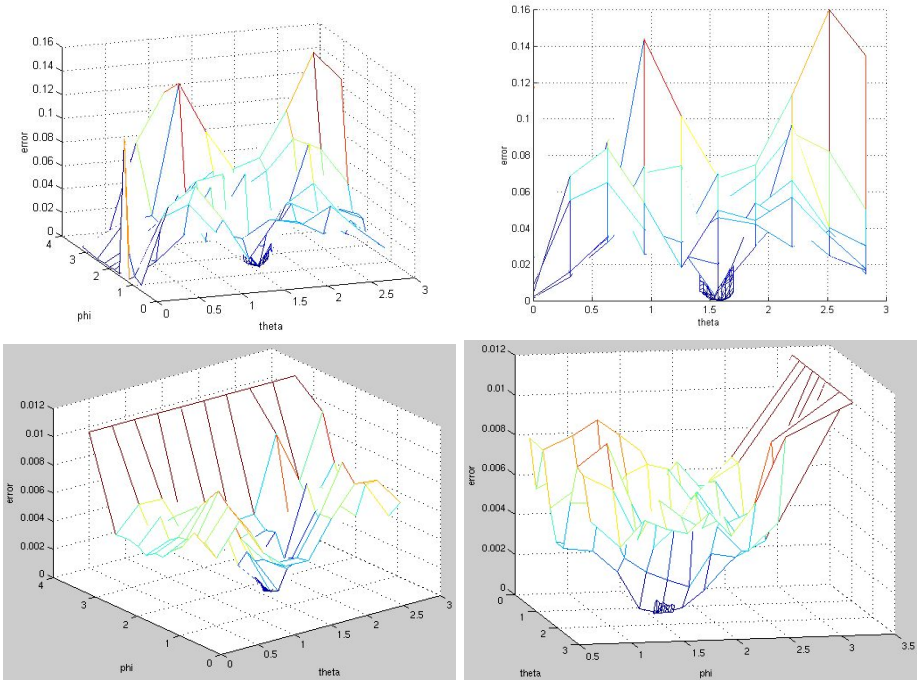


Figure 8.12: Quality number of axis orientation as a function of θ and ϕ. Top: using minimum bounding box of circle centres; below: using the difference between radii and distances {circle centre} to {plane-mesh intersection}.

Now the entire algorithm has been explained, we look in more detail into the possibilities to measure the correctness of the tested orientation:

- If the tested orientation was the correct orientation, the resulting circle centres should be collinear on a line orientated in the same way. Hence the x and y coordinates of the estimated circle centres should coincide after the circles have been projected onto a plane perpendicular to the chosen orientation. One can use the **area** of the smallest **bounding box** in that plane containing all circle centres. This approach is cheap: it only requires $O(P)$ operations which can be neglected compared to the $O(V)$ of the entire evaluation algorithm. If the quality number is plotted as a function of θ and ϕ, it can be seen that this approach has local minima that almost compete with the global minimum. The error function can be seen in the upper half of figure 8.12. The orientation selected is the correct one, but other orientations have quality numbers that are only slightly

bigger. Figure 8.13 displays the mesh rotated over $-\theta$ around the Z axis and $-\phi$ around the Y axis (rotate (θ, ϕ) back to the Z axis). The results of our algorithm are also plotted: the mesh points in black, the intersection points with the P planes in green, the estimated circles in red. The circle centres are in blue, and the corresponding bounding box in gray. For both figures, the area of the bounding box is small: they visualise two of the local minima (that are larger than the global one) in the (θ, ϕ)-space for the "bounding box" error function.

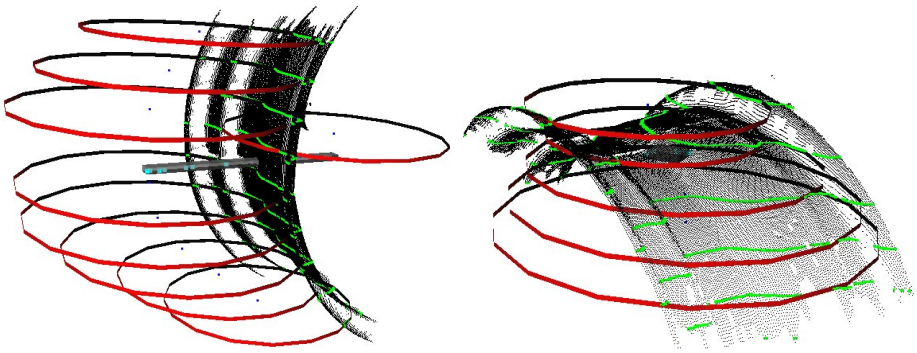


FIGURE 8.13: Erroneous solutions corresponding to local minima of the error function as shown in figure 8.12: left: local minimum for the "bounding box" approach, right: for the "circle centre distance" approach.

- Note that the bounding box on the left of figure 8.13 is elongated. The corresponding local minimum can be resolved (increased) by not only considering the area of the bounding box, but also incorporating the **maximum length** of either side of the bounding box into the quality number. This is a upper bound for the maximum distance between any two circle centres. For the local minimum on the right hand side of figure 8.13, this is not a solution, as it has near collinear circle centres along the chosen orientation.

- The planes perpendicular to the rotational axis intersect with the mesh in a number of points. The **distance** from those points **to** the **circle centres** just estimated, are a better cue to the quality of the fit. Average out the absolute value of the difference between this distance and the corresponding circle radius, over all the intersection points in a circle, and over all circles. This uses $O(\sqrt{V})$ flops, more that the previous two, but still negligible compared to the entire algorithm, which is $O(V)$
  As can be seen in figure 8.12, the global minimum is more pronounced and local minima are less pronounced than in the "bounding box" approach. Hence, the results plotted in figure 8.9 and 8.10 use this approach.

Generatrix accuracy test

In order to test the correctness of the axis estimate, calculate the distance of every vertex to the axis, and plot the results in 2D (left of fig.8.14): the distances horizontally, the position along the axis vertically. For a analytical expression of the generatrix one needs to fit a curve through those points. The thickness of the bounds of the generatrix points gives an indication of the accuracy. Calculating this thickness as a function of the position along the axis yields values of about 5 to 7 mm, thus the maximum difference with the fitted curve is about 3 mm. Automated removal of the burr will probably require other sensors than vision alone. For example, when the end effector of the robot arm touches the burr, the removal itself is better of also using the cues of a force sensor. Therefore, a fusion of sensory inputs – at different frequencies – is needed. Because one needs different sensors at close range anyway, a vision based accuracy of about 3 mm is enough. The right side of fig.8.14 shows the mesh plotted in cylindrical coordinates: the 'dist' axis is the distance from each point to the axis of the surface of revolution, the 'angle' axis is the angle each point has in a plane perpendicular to the axis of the surface of revolution and the 'axis' axis is the distance along the axis of the surface of revolution. In green are the estimated circles in the same coordinate system. The data is almost constant along the 'angle' axis, as expected.

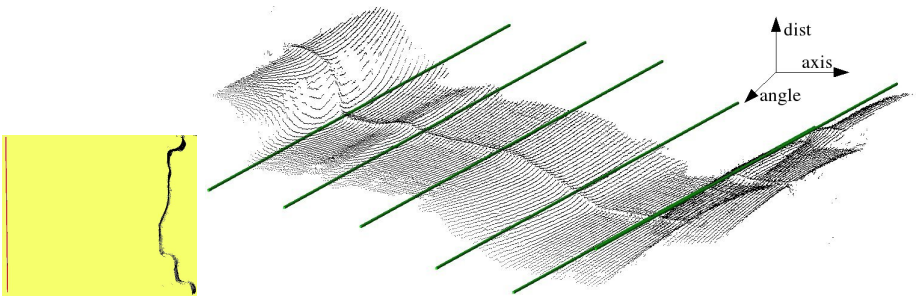


Figure 8.14: Left: axis and generatrix, right: distances to the axis: the burr is clearly visible.

8.3.6 Conclusion

This experiment presents a robust technique for the detection and localisation of flaws in industrial workpieces. More specific it contributes to burr detection on surfaces of revolution. As the object is metallic, the structured light was adapted to compensate for the infamous specularity problem. This is done using adaptive structured light.
The algorithm runs in three steps. First the orientation of the axis is extracted from the scan data. Secondly the axis is localised in space. Finally these parameters are used to extract the generatrix of the surface. The data is then compared to this generatrix to detect the burr.
The use of RANSAC estimation renders the algorithm robust against outliers. The use of particle filtering is also discussed. Next to this the system explicitly checks if our solution is correct by the application of a set of secondary tests. Correctness of the detection is important as this data is to be used for robotic control.

8.4 Automation of a surgical tool

8.4.1 Introduction

This section presents a surgical application of structured light in robotics. It is an example of the **integration of 2D and 3D** — structured light — **visual cues**, see section 6.3.
The organs of the abdomen are the objects of interest here, as it is another example where both 2D and 3D vision is not evident. Organs also suffer from specular reflections, as was the case for the previous wheel deburring experiment (section 8.3). They also have **little visual features**. The latter makes them a good candidate for artificially creating visual features through structured light (see section 3.2.1).



Figure 8.15: Unmodified Endostitch

More in detail, this section describes the automation of a surgical instrument called the Endostitch. It is a manual **tool for laparoscopic suturing**, see figure 8.15. Laparoscopy refers to endoscopic operations within the abdomen. The goal of this research is to semi automate suturing after specific laparoscopic procedures using the automated laparoscopic tool mounted on a robotic arm.

Research motivation

These operations usually only make four incisions of a few mm: two for surgical instruments, one for the endoscope and one for insufflating CO_2-gas in order to give the surgeon some work space. These incisions are a lot smaller than the incision that is made for open surgery, which is typically $20cm$ long. The advantages are clear: less blood loss, faster recovery, less scar tissue and less pain (although the effects of the gas can also be painful after the operation).
However, performing these operations requires specific training, they are more difficult since the instruments make contra-intuitive movements: e.g. moving one end of the instrument to the left moves the other to the right, as the instruments have to be moved around an invariant point. The velocity of the tool tip is also scaled depending on the ratio of the part of the instrument out and inside the body at every moment. This can amplify the surgeon's tremor. Other matters that make this task difficult for the surgeon is the reduced view on the organ, and the lack of haptic feedback. The organ cannot be touched directly as is often useful, only felt through the forces on the endoscopic instruments.

Suturing at the end of an endoscopic operation is a time demanding and repetitive job. The faster it can be done the shorter the operation, and the faster the patient can recover. Moreover, robotic arms are more precise than the slightly trembling hand of the surgeon. Therefore, it is useful to research (semi)automate suturing.

Visual control

This experiment **automates** an (otherwise manual) Endostitch and controls it using a digital I/O card in a PC. To this end a partial 3D scan of a (mock-up for an) organ is acquired using structured light. 2D and 3D vision algorithms are combined to track the location of the tissue to be sutured. Displaying a 3D reconstruction of the organs eases the task of the surgeon since the depth cannot be guessed from only video images. Another reason for calculating the field of view in three dimensions is its need for estimating the motion of the organ (and then compensating for it in the control of the robot arm), or extracting useful features.

Several approaches have been explored to **measure** the **depth** in this setting:

- Stereo endoscopes have two optical channels and thus have a larger diameter than normal endoscopes. Therefore often not used in practice, other types of 3D vision need to be explored for this application:
- Thormaehlen et al. [2002] presented a system for three-dimensional endoscopy using optical flow, with a reconstruction of the 3D surface of the organ. For this research he only used the video sequences of endoscopic surgery (no active illumination is needed for structure from motion).
- Others use laser to actively illuminate the scene, like Hayashibe and Nakamura [2001] who insert an extra instrument into the abdomen with a laser. That laser scans the surface using a Galvano scanner with two mirrors, and triangulation between the endoscope and the laser yields a partial 3D reconstruction of the organ.
- The group of de Mathelin also inserts such an extra laser instrument: Krupa et al. [2002] use a tool that has three LEDs on it and projects four laser spots onto the organ. This limited structured light method enables them to keep calculations light enough to do visual servoing at high frequencies (in this case at $500Hz$), and allows to accurately estimate repetitive organ motions, as for example published by Ginhoux et al. [2003].

The remainder of this experiment will study a structured light approach using a normal camera and LCD or DLP projector. Section 3.2.2 describes what types of projectors may be better suited for this application. However, the same algorithms apply. If one wants to combine 2D and 3D vision, the projector has to constantly switch between full illumination and structured light projection. In an industrial setting, this **stroboscopic** effect needs to be avoided, as it is annoying for the people that work with the setup. However, inside the abdomen,

this is not an issue. Even if we would like to only use structured light, in practice the system would still have to switch between full illumination and structured light, as the surgeon wants to be able to see the normal 2D image anyhow. So the system will switch the light, and separate the frames of the camera to canalise them to the normal 2D monitor on one hand, and the 3D reconstruction on the other hand.

This experiment is organised as follows: section 8.4.2 elaborates on the automation of the laparoscopic tool and section 8.4.3 discusses the work that has been done on the robotic arm motion control. Section 8.4.4 gives an overview of the structured light scanning and section 8.4.5 explains the combination of the 2D and 3D vision.

8.4.2 Actuation of the tool

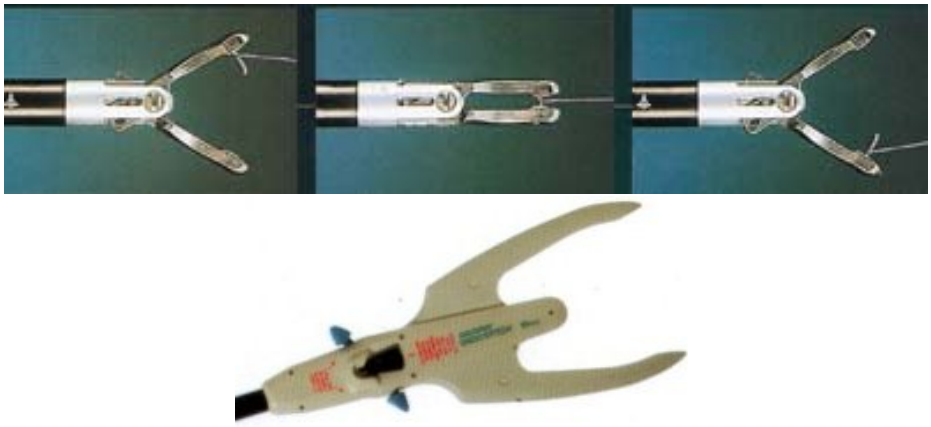


Figure 8.16: Detailed view of the gripper and handle of an Endostitch

An Endostitch is a suturing tool developed by Auto Suture in 2003 that decreases the operative time for internal suturing during laparoscopic surgery. It has two jaws, a tiny needle is held in one jaw and can be passed to the other jaw by closing the handles and flipping the toggle levers. It is mainly used for treating hernias and in urology, and also for suturing of lung wounds inflicted by gunshots for example.

Type of actuation

This tool was automated **pneumatically** with a rotative cylinder around the toggle levers and a linear one for the handles. Both cylinders have two end position interrogation sensors at either end of their range. Those sensors are electric proximity switches that output a different voltage whether or not they are activated. We connect those signals to the inputs of a digital I/O card. The state the laparoscopic tool is in can thus be read in software. Note that the

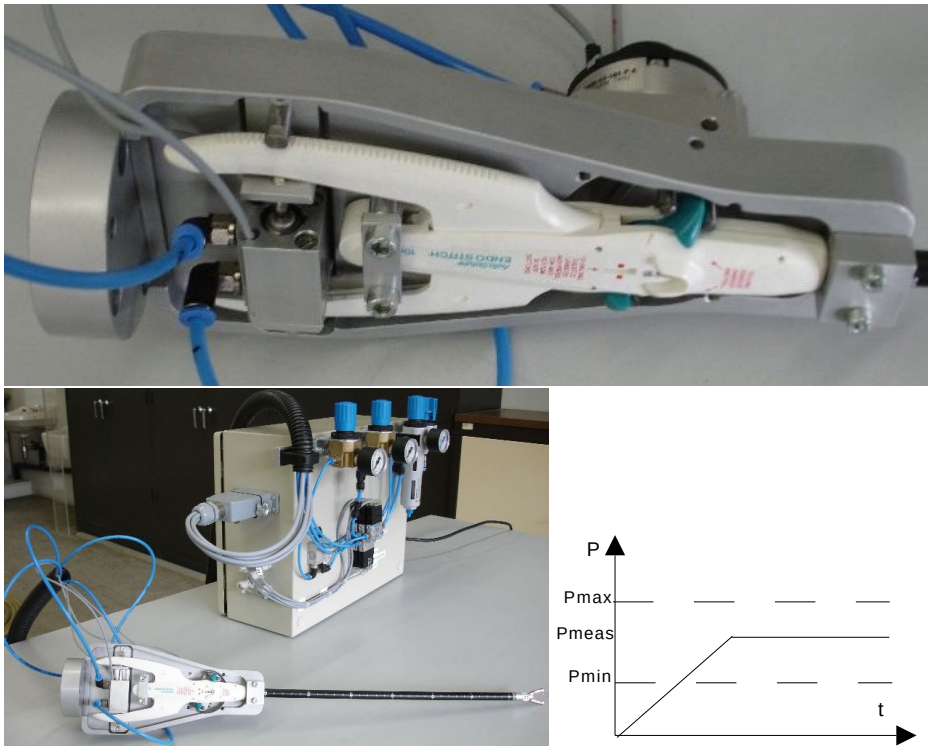


FIGURE 8.17: Top and bottom left: pneumatically actuated Endostitch, bottom right: two possible pressures

laparoscopic tool could also have been actuated using electrical motors, as Göpel et al. [2006] do. Both systems have a **decoupling** between the power supply and the tool at the robot end effector. In the pneumatic case, the power supply is the differential air pressurisation, that remains in a fixed position near the robot (see the bottom left picture of figure 8.17). In the **electric** actuation case, the motor is also in a fixed position near the robot, and Bowden cables are used to transmit the forces to the tool at the end effector. The need for this mechanical decoupling is similar to the need for the optical decoupling for laser projectors, see section 3.2.2. The powering sources for both mechanical and optical systems are generally to heavy, large or fragile to attach them rigidly to the end effector.

Interfacing

The pneumatic cylinders are actuated using TTL logic on the same card. The actuation makes use of the **scripting** language of our robot control software Orocos [Bruyninckx, 2001] to implement a program that stitches any soft material, it currently functions at $2Hz$ (a C^{++}-interface is also provided). It was implemented as a **state machine**, see the left hand side of figure 8.18, and

section 7.2.4 for more general information on these state machines. The each of the cylinders can have three states: actuated in either direction or not actuated.

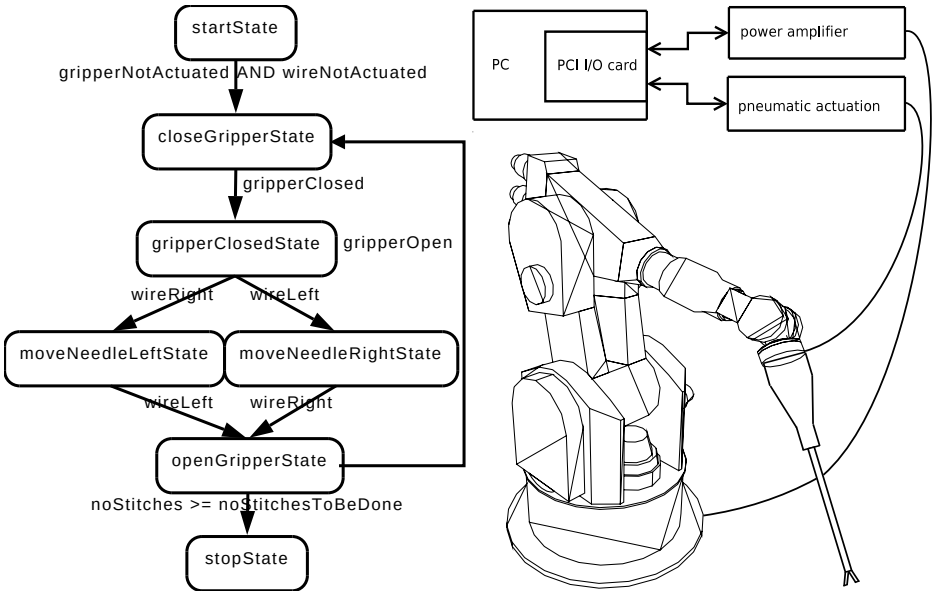


Figure 8.18: Left: state machine, right: setup with robot and laparoscopic tool

Anatomical need for force differentiation

The local quality of human tissue determines whether it will hold or might tear when stressed after the operation. Therefore it is important to stitch using the stronger parts of the tissue. When trying to close the jaws with tissue in between, the pressure could be increased linearly until the needle is able to pass through. That minimal pressure needed (P_{meas}) is a relevant cue to the quality of the tissue, and hence to whether or not that spot is suitable for the next stitch.
Experiments identified a minimal pressure P_{min} below which the tissue is not considered to be strong enough, and a maximal pressure P_{max} above which the tissue is probably of a different type. If the pressure needed to penetrate the tissue is in between those two, the stitch can be made at that location, see figure 8.17.

However, since it's only important that $P_{min} \leq P_{meas} \leq P_{max}$, then it is faster only to use the **2 discrete pressures** P_{min} and P_{max} instead of continuously increasing the pressure. First try the tissue using P_{min}, if it passes through the tissue, the tissue is not strong enough. If it doesn't pass through the tissue, try again using P_{max}. If it passes this time, the tissue is suitable, otherwise try another spot.
As can be seen in figure 8.17, the case of the pneumatic actuation has three

adjustable valves (in blue on top), that is one master valve that can decrease the incoming pressure, and two valves for further reduction to P_{min} and P_{max}. Switching between P_{min} and P_{max} can be done in software.

8.4.3 Robotic arm control

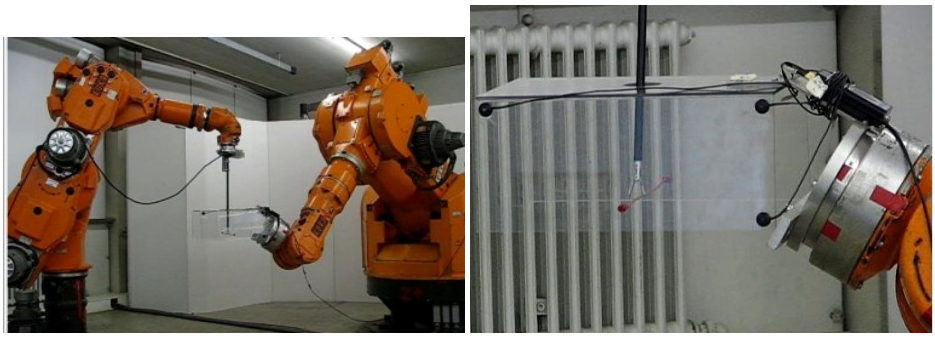


FIGURE 8.19: Left: 2 Kuka robots cooperating on minimally invasive surgery mock-up, right: closeup with three LEDs

The automated laparoscopic tool is mounted onto a robotic arm as schematically illustrated on the right hand side of figure 8.18. To test this functionality individually, this visual servoing does not use structured light, but a Krypton K600 CMM: a vision system that tracks LEDs attached to an object. To that end, it uses three line cameras, and derives the 6D position of the object from it.
Figure 8.19 shows how one robotic arm holds the mock-up for the abdomen. It has a hole in it, simulating the trocar point [2]. A 3D mouse moves this mock-up in 3D, simulating the motion of the patient (e.g. of respiration).
The other robotic arm makes the gripper of the laparoscopic tool move along a line (e.g. an artery) inside our 'abdomen' while not touching the edges of the trocar and compensating for the motion of the 'patient'. The robotic arm has six degrees of freedom, which are reduced to four in this application, since the trocar point remains constant. This setup is an application of the **constraint based task specification** formalism [De Schutter et al., 2005], for more details see [Rutgeerts, 2007].

It is interesting to replace this vision system by a structured light system, as attaching LEDs to organs is clinically difficult to impossible. Also the assumption that the rest of the observed body is rigidly attached to it is not valid: one would have to make an estimation of the **deformation** of the tissue.

[2] the small incision in the abdomen

8.4.4 Structured light depth estimation

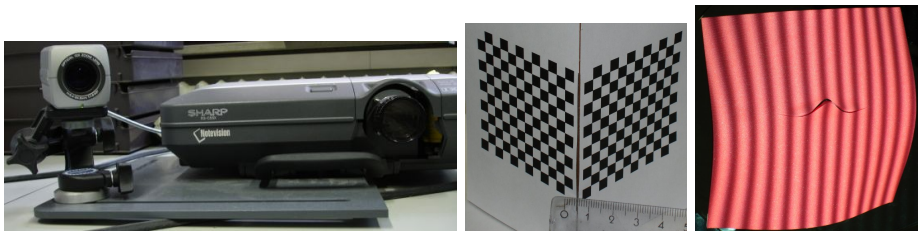


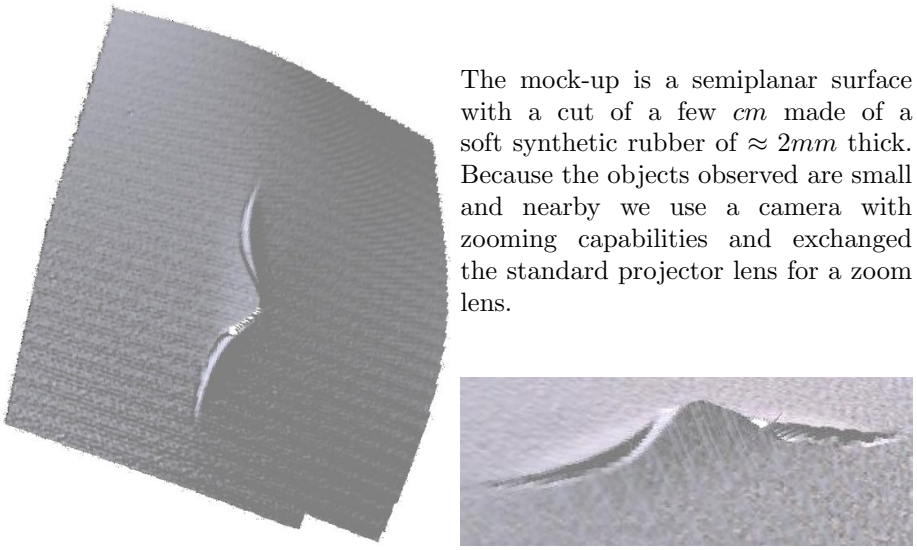
Figure 8.20: Setup with camera, projector, calibration box and mock-up

Hardware difficulty

In laparoscopy the structured light needs to be guided through the endoscope. The percentage of the **light** that is **lost** in sending it through the fiber is considerable: about half of the light is absorbed. In addition to that it is interesting a use a small aperture for the endoscope: the smaller the aperture, the larger the depth of field. Then the image remains focused over a larger range, but the light output is reduced further. However, also the projector has a limited depth of field, and it is not useful to try and increase the camera depth of field far beyond the projector depth of field as both devices need to be used in the abdomen. Still, summarising, one needs a more **powerful** and thus more expensive **light source** for structured light in a laparoscopic context (an order of magnitude 5000 lumen projector is needed). Armbruster and Scheffler [1998] built a prototype whose light intensity in the abdomen is sufficient. This thesis decided not to build such demanding prototype, but to perform the first feasibility study using a standard camera and projector, see figure 8.22. A laser projector may be a better choice here, see section 3.2.2.

Different structured light approaches

For this experiment, as for the previous one (section 8.3), the 3D feature to be detected is relatively **subtle**. Therefore we use the same **high resolution** 1D Gray code structured light implementation, augmented with phase shifting. The disadvantage of this approach is again that most of the fine 3D data is discarded, and only a very small fraction of it is useful. Therefore, a lot of meaningless computations are done, a situation to avoid since the **bottleneck** of these structured light systems is the **computational load**. Therefore, it is interesting to apply the sparse 2D spatial neighbourhood structured light technique of section 3.3.6 here too. The difference with the previous experiment is that there is no 3D model at hand here. However, one has the 2D visual input as a base.



The mock-up is a semiplanar surface with a cut of a few *cm* made of a soft synthetic rubber of $\approx 2mm$ thick. Because the objects observed are small and nearby we use a camera with zooming capabilities and exchanged the standard projector lens for a zoom lens.

FIGURE 8.21: High resolution reconstructions for static scene

Thus a possible strategy in this case is:

- First pattern: the **sparse 2D pattern** discussed throughout this thesis, see the top left part of figure 8.22. This returns a sparse 3D reconstruction, and the corresponding 2D projector coordinates.
- Second pattern: full illumination (bottom left part of figure 8.22): send this image to the normal monitor for the surgeon, and possibly use it to extract extra 2D vision information (see section 8.4.5).
- Third pattern: a series of **line segments** approximately perpendicular to the wound edge (or other organ part of interest), where more 3D detail is desired. Pattern encoding is not necessary, as the correspondence can be deducted from the sparse 2D-3D correspondences just calculated (spectral encoding is anyhow not useful here, as organs are coloured). This is a form of active sensing: the missing information is actively detected based on previous information.

These patterns are repeated **cyclicly**, preferably several times a second to keep track of the state the suturing is in. An important asset of this technique is that it is capable of working with such a dynamical scene, while the 1D Gray coded structured light is limited to a static scene.
Organs cause specular reflection, just like the metal wheel of the previous experiment does. On these highlights the camera normally oversaturates, for which the software compensates, see section 8.3 for details.

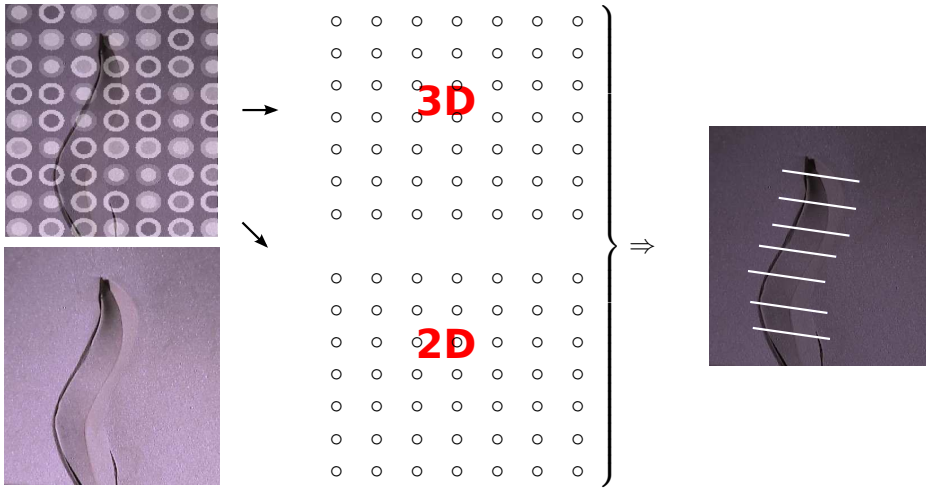


Figure 8.22: Structured light adapted for endoscopy

8.4.5 2D and 3D vision combined

If a robot uses multiple sensors to move in an uncertain world, this sensory information needs to be merged. However, every *sensor* does not need to be a physically different device. A camera for example can be subdivided in several **subsensors**: 2D edge data, 2D colour data, depth data are three of those possible subsensors. Hence, the experiment uses several (2D and 3D) visual cues based on the same images to enhance the robustness of the localisation and tracking of the incision edges.

2D wound edge detection

To determine the region of interest, **2D** vision techniques are used. The surgeon uses a mouse to roughly indicate a few points of the part to be sutured. A **spline** is fitted through these points and an edges are detected along line segments perpendicular to the knot points indicated. We use a 1D exponential filter to detect intensity edges larger than a certain threshold: it is the **ISEF filter** by Shen and Castan [1992], see the white dots in figure 8.23 on the top left. To track this curve, this implementation uses a active contour, or **snake** [Blake and Isard, 1998].

3D wound edge detection

In 3D a similar fitting can be done roughly modelling the cross section of the wound could as two Gaussians, since the tissue tension pulls the wound edges apart (see the bottom right graph of figure 8.23.
The 2D knot pixels are used as texture coordinates to fit a **3D spline** through

the corresponding mesh points. A similar active contour algorithm is used, now in 3D: the edges are searched in the intersection between the mesh and a filled circle around the knot point perpendicular to the spline, see the bottom right picture of figure 8.23. An example of actual intersection data can be seen in the bottom left graphic of the same figure. The data can be locally noisy (small local maxima), and outliers need to be avoided in visual robot control. Therefore, the robustness of the method to extract the two maxima is important: an **Expectation-Maximisation** approximation is used to reduce noise sensitivity [Dempster et al., 1977].
Inverse transform sampling is applied to the 1D signals on the bottom left of figure 8.23, because one needs probabilistically sound samples of the distance perpendicular to the wound edge, along the overall tissue direction. In other words the input of the EM algorithm should be the pixel values along the horizontal axis of the graph, not of the height values on the vertical axis [Devroye, 1985]. EM can be used here since we know that we are looking for a certain fixed number of peaks, in this case two. Having a reasonable estimate of the initial values of the peaks is necessary. Choose one to be the maximum of the intersection and the other one symmetrical around the centre of the intersection, as indicated by the vertical lines in the graph on the bottom of figure 8.23.

Computational complexity

As this system functions online, it needs to be time efficient. Only the line segments (for the 2D image) or filled 3D circles perpendicular (for the mesh) to the 3D spline are searched. The data are 1D intensity values for the former case, and 1D height values in the latter case. Within this drastically reduced search spaces, we use efficient algorithms: the 2D edge detector is $O(N_0)$ where N_0 is the number of intensity values in the line segment. Inversion sampling is also $O(N_1)$ where N_1 is the number of height values in the mesh intersection. The EM algorithm is iterative but ≈ 5 iterations are sufficient for a reasonable convergence. Its complexity is $O(I \cdot C \cdot N_2)$ where I is the number of iterations, C is the number of classes (two in this case) and N_2 the number of height values in the intersection. As I and C are constant parameters, the EM procedure is $O(N_2)$, and thus also **linear**.

Improvements and extensions

Possible improvements to increase the robustness include: a **voting** scheme (Bayesian or not) based on the 2D and 3D estimation to improve the tracking, and adding other 2D (or 3D) cues in this voting scheme.
If one does not look for wound edges, but other anatomic features that are not even approximately Gaussian shaped, the technique can still be used. In that case, **3D shape descriptors** are useful [Vranic and Saupe, 2001]. These descriptors are similar to the 2D ones used in section 3.3.5, as they are also based on a Fourier transform. They summarise a shape in a few characterising numbers, that are independent of translation, rotation, scaling, (mathematical) reflection and level-of-detail.

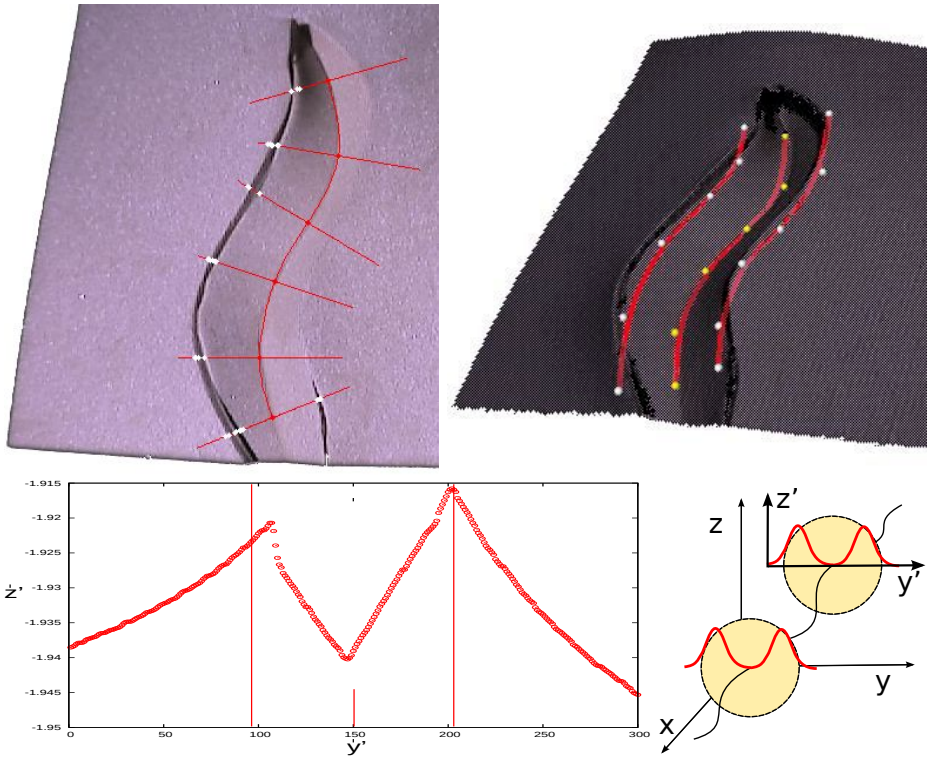


Figure 8.23: Top: results in 2D and 3D for statical reconstruction, bottom left: intersection with a plane perpendicular to the fitted spline, bottom right: Gaussian model of the wound edges

8.4.6 Conclusion

This experiment presents the automation of a suturing tool. The tool is attached to a robotic arm of which the motion is the sum of the desired motion for the surgical procedure and a motion compensation of (a mock-up of) the patients abdomen. To track the deformable incision in the organ we use a combination of 2D (snakes) and 3D vision (3D snakes using EM estimation).

8.5 Conclusion

The last experiment of this chapter showed the broad usefulness of the proposed 2D structured light technique to estimate the depth at a large number of points to an arbitrary object in the scene, and control a robot arm with it. However, the first two experiments in this chapter introduce concrete applications that cannot be solved with this technique alone. Both the industrial experiment, and the surgical one benefit from both the sparse 2D type of structured light, and a type of structured light that is adapted to the task at hand.

Chapter 9

Conclusions

9.1 Structured light adapted to robot control

This thesis describes the design and use of a 3D sensor adapted to control the motion of a robot arm. Apart from 2D visual information, **estimating the depth** of objects in the environment of the robot is an important source of information to control the robot. Applications include deburring; robot assisted surgery and welding, painting and other types of object manipulation. To estimate the depth this thesis applies stereo vision to a camera-projector pair.

Choosing a structured light technique (see section 2.2.2) is always a **balance between speed and robustness**. On one side of the spectrum are the single stripe methods (using a moving plane of laser light): those are slow since they need an image frame to reconstruct each of the intersections between the laser plane and the scene while the laser is moving. But they are robust as there can be no confusion with other lines in the image.
On the other side of the spectrum are densely coded single frame methods: fast but fragile. Multi-stripe, multi-frame techniques are in between: they use less frames than the single stripe methods and are relatively robust.

Another **balance is between resolution and robustness**: the larger one makes the image features, the more certain it is they will be perceived correctly. This work chooses a position on those balances that is most suitable for controlling the motion of a robot arm: it presents a method that is *fast* and robust, but low in resolution. *Fast* is to be interpreted as *single shot* here: it allows to work with a moving scene. The robustness is ensured

- by not using colours, so there is no need for an extra image frame to do colour calibrations or to adapt the colours of the pattern to the scene. With such an extra frame the technique wouldn't be single shot anymore.
- by using sufficiently large image features: this makes decoding them easier.
- by using a very course discretisation in intensity values: the larger the visual difference between the shades of grey, the clearer it is how the features

should be decoded. The system can use even fewer projected intensities than would normally be the case, as a projection feature is composed of two intensities (for reasons explained in the next point). The features then become larger, and therefore the resolution lower. We are however willing to pay the price of a low resolution as the application the structured light is applied to here, a robotic arm, is one where the geometrical knowledge about the scene can be gradually improved as the end effector (and thus the camera) is moving closer to the scene. The combination of all these coarse reconstructions using e.g. Bayesian filtering, allows for sufficient scene knowledge to perform many robot tasks.

- by setting one of the two intensity values in each feature to a fixed (maximum) projector value. This way one can avoid the need to estimate different reflectivities in different parts of the scene, due to the geometry or the colour of the scene. Thus, the technique decodes intensity levels relatively instead of absolutely.

Figure 9.1 presents the different processing steps in the vision pipeline. The encoding chapter, chapter 3, discussed the first 3 elements on the top left of the figure: pattern logic, implementation and adaptation. This decoding chapter, chapter 5 discussed the 2 blocks below: the segmentation and labelling procedures. Before one can reconstruct the scene, some parameters have to be estimated: the calibration chapter, chapter 4, explained all of these necessary calibrations, shown on the right hand side of figure 9.1. Chapter 5 ends with the actual 3D reconstruction. We now **discuss the robustness of each of these steps**.

- Pattern logic, implementation, and adaptation are exact steps, and have no robustness issues.

- Segmentation is often problematic because of fixed thesholds. Therefore, we do not use such thresholds. At every step where they are needed, they are not hard coded, but estimated based on the sensory input to increase the robustness.

- Labelling is a step that could be limited to simply detecting the nearest neighbours that are needed to decode each blob. However, to increase the robustness of this step, this step performs a consistency check to make sure all neighbours in the grid are reciprocal.

- On the right side of the figure, the camera and projector intensity calibrations are straightforward identification procedures based on overdetermined systems of equations. Also the hand-eye calibration and the lens aberration compensations are stable techniques without robustness problems.

- The geometric calibration however needs precautions to avoid stability problems. In order to make it more robust, it uses – wherever possible –

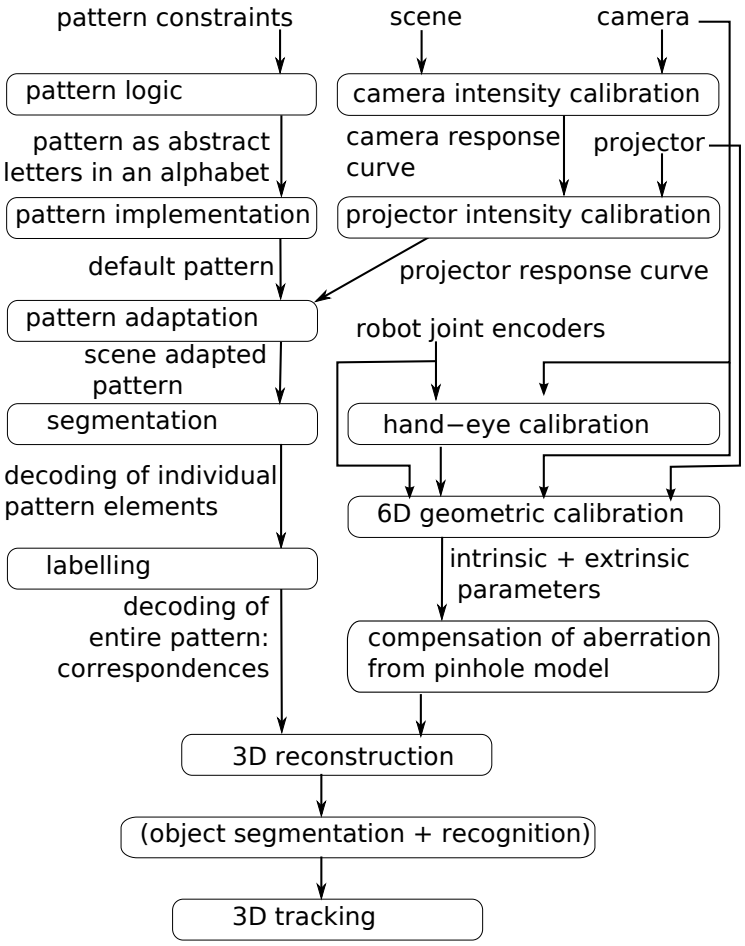


FIGURE 9.1: Overview of different processing steps in this thesis

the known motion of the camera through the encoder values of the robot joints. Section 4.4.4 explains how this technique works both with planar and non-planar surfaces.

- The actual 3D reconstruction is a relatively simple step, using a known stable technique.

9.2 Main contributions

Of the contributions of the introductory chapter the most important are :

- This thesis presents a new method to **generate** 2D spatial neighbourhood **patterns** given certain desired properties: their window size, the minimal Hamming distance and the number of letters in the alphabet. These

patterns can be made independent of their orientation such that no extra processing needs to be done online to determine which side of the camera corresponds to which side of the projector image. This method bases itself on established brute force techniques, but orders the search in the search space such that better results are obtained. The resulting patterns are compared to existing techniques: for constant values of the desired properties, they are larger; or for a constant size, they have superior properties. Section 3.2.5 explains the algorithm, and appendix A contains its pseudo-code, such that its results are reproducible.

- Section 3.3 explains the different graphical manners to put this pattern logic into practice in the projector image. It introduces a new pattern that does not require the scene to have a near white colour. It avoids segmentation problems due to locally different reflectivity, by not relying on absolute intensity values, but rather on a local **relative** comparison between **intensity levels**. It needs only one camera frame to extract the range information.
- Section 6 demonstrates that this sensor can seamlessly be **integrated into** existing methods for **robot control**. *Constraint based task specification* for example allows to position the end effector with respect to the projected blob, while possibly also taking into account motion constraints from other sensors.

9.3 Critical reflections

Section 5.2.4 explained under what conditions the **structured light sensor fails**. Some of these failure modes are very unlikely to happen in practical situations: failing when the scene is so far away that the illumination is too weak, and the condition when external light sources are similar to the projected pattern. The other failure modes are more likely to happen:

- When the scene is **too far** away **in comparison to the baseline**. This happens when the end effector moved close to the projector.
- When the scene is **not locally planar**. For example, if one attemps to execute a robot task on a grating with hole sizes that are in the range of the size of the projected blobs. Large parts of all the blobs will then not reflect, and make decoding impossible.
- Structured light is a technique that removes the need for scene texturing to be able to solve the correspondence problem, and thus reconstruct the depth of points in the scene. On the other hand, if the scene is **highly** textured, it may fail. In other words: if locally, withing a feature projection, reflectivity is very varied, the range estimation may fail.
- In case of **self occlusion**: when the robot moves in between the projector and the scene all communication between projector and camera is cut off.

The structured light method presented in section 3.3.6 is designed to be broadly applicable, for cases where no additional **model information** is available. However, where this information is available, it is better to use it. This is a form of **active sensing**: actively adapting the projection according to where additional information is needed. The experiment in section 8.3 is an example of this.

Another important point when constructing a structured light setup for a concrete robot task, is the relative positioning between camera and projector. Section 3.2.2 explains how this positioning is hardware dependent. If the setup permits a **rigid transformation between camera and projector frames** (with LED or laser projectors for example), the complexity of the geometric calibration can be reduced.

This thesis discusses only one robot sensor. In order to reliably perform robot tasks, it is important to **integrate the information from different sensors**. Each sensor can address the shortcomings of another sensor. It is often far better to use information from different more rudimentory sensors, than to use only one more refined sensor.
The field of range sensors has evolved in recent years: section 2.2 explains the development of short range (order of magniture $1m$) time-of-flight sensors. The integration of such sensor with vision based on natural features seems a promising reasearch path.

9.4 Future work

This work does not implement all elements of the graphics pipeline presented in figure 9.1. Some elements remain :

- **3D tracking**: Section 5.2.3 describes how to track the 2D image features. Since tracking is the restriction of a global search to a local one, one should try and track as much as possible for efficiency reasons.
  It is also possible to track 3D features: following clusters of 3D points based on their curvature (see for example [Adán et al., 2005]). In order to calculate the curvature one needs a tessellation of the points, which is relatively easy in this case, as one can base it on the labelling procedure of section 5.3. Tracking in 2D and 3D can support each other:
  - If all 2D image features could always be followed, 3D tracking would be nothing but a straightforward application of triangulation formula on the 2D tracking result. However, features often get occluded or deformed.
  - Section 3.4 decided to adapt the size and position of the features in the image if desirable: also in that case 2D tracking becomes difficult.

  As we are dealing with an online feed of measurements giving data about a relatively constant environment, Bayesian filters are useful to integrate this 4D data (3 spatial dimensions and the time).

- **Object clustering and recognition**: the result of the previous step in the pipeline is a mesh in motion. This mesh is the envelope of all objects in the scene. Clustering it is an important step to recognise individual objects. After (or more likely simultaneously with) clustering, one classifies the objects according to a database of known objects. Integrating several cues makes this process easier: it is better to use not only the mesh, but also the 2D information in the images: edges, colours ... The camera used to reconstruct the projector image has an undersaturated background, so a second camera would have to be used to extract this 2D information, or the shutter time of the camera would have to be switched frequently. Two approaches for classification of objects exist:

  - Model based: simply comparing object models with real objects or collections of real objects has proved to be a difficult path.
  - Subspace methods like PCA and LDA: these produce promising results. Blanz and Vetter [2003] for example uses PCA to span a **face space**. The algorithm starts with the automatic recognition of control points. Using several hundred test objects (in this case faces), an eigenspace can be spanned that can recognise objects with higher success rates than the standard model based approach.

  Taking this clustered mesh into account while tracking 3D features, improves the tracking as one knows which features belong together.

- Near the end of this work, new technology emerged that is interesting in this application domain. More precisely, following existing technologies improved to the level of becoming useful in the field of range sensing for robot arm control: LED projectors, laser projectors and Lidar range scanners. Section 3.2.2 and 2.2 discuss the advantages and disadvantages of each of these technologies. It would be interesting to design experiments to compare these systems to the triangulation technique discussed in this thesis, in the context of robotics.

Bibliography

Adams, D. (1999). *Sensor Modelling, Design and Data Processing for Autonomous Navigation*. World Scientific Series in Robotics and Intelligents Systems, Vol 13, ISBN: 981-02-3496-1.

Adán, A., F. Molina, and L. Morena (2004). Disordered patterns projection for 3d motion recovering. In *International Symposium on 3D data processing, visualization, and transmission*, pp. 262–269.

Adán, A., F. Molina, A. Vásquez, and L. Morena (2005). 3d feature tracking using a dynamic structed light system. In *Canadian Conference on Computer and Robot Vision*, pp. 168–175.

Aloimonos, Y. and M. Swain (1988). Shape from texture. *BioCyber 58*(5), 345–360.

Andreff, N., R. Horaud, and B. Espiau (2001). Robot hand-eye calibration using structure-from-motion. *The International Journal of Robotics Research 20*(3), 228–248.

Armbruster, K. and M. Scheffler (1998). Messendes 3d-endoskop. *Horizonte 12*, 15–6.

Benhimane, S. and E. Malis (2007, July). Homography-based 2d visual tracking and servoing. *International Journal of Robotic Research 26*, 661–676.

Blais, F. (2004). Review of 20 years of range sensor development. *Journal of Electronic Imaging 13*(1), 231–240.

Blake, A. and M. Isard (1998). *Active Contours*. Berlin, Germany: Springer.

Blanz, V. and T. Vetter (2003). Face recognition based on fitting a 3d morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(9), 1063–1074.

Bouguet, J.-Y. (1999). Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. Technical report, Intel Corporation.

Bradski, G. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel technology journal 1*, 1–15.

Brown, D. (1971). Close-range camera calibration. *Photogrammetric Engineering 37*, 855–866.

Bruyninckx, H. (2001). Open RObot COntrol Software. http://www.orocos.org/.

Caspi, D., N. Kiryati, and J. Shamir (1998). Range imaging with adaptive color structured light. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*(5), 470–480.

Chang, C. and S. Chatterjee (1992, October). Quantization error analysis in stereo vision. In *Conference on Signals, Systems and Computers*, Volume 2, pp. 1037–1041.

Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G. . Hager, and A. Morse (Eds.), *The Confluence of Vision and Control*, pp. 66–78. LNCIS Series, No 237, Springer-Verlag.

Chen, S. and Y. Li (2003). Self-recalibration of a colour-encoded light system for automated three-dimensional measurements. *Measurement Science and Technology 14*, 33–40.

Chen, S., Y. Li, and J. Zhang (2007, April). Realtime structured light vision with the principle of unique color codes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 429–434.

Chernov, N. and C. Lesort (2003). Least squares fitting of circles and lines. *Computer Research Repository cs.CV/0301001*, 1–26.

Claes, K., T. Koninckx, and H. Bruyninckx (2005). Automatic burr detection on surfaces of revolution based on adaptive 3D scanning. In *5th International Conference on 3D Digital Imaging and Modeling*, pp. 212–219. IEEE Computer Society.

Curless, B. and M. Levoy (1995). Better optical triangulation through spacetime analysis. In *Proceedings of the 5th International Conference on Computer Vision*, Boston, USA, pp. 987–994.

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research 18*(3), 286–298.

Davison, A. (2003, October). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision*.

De Boer, H. (2007). Porting the xenomai real-time firewire driver to rtai. Technical Report 024CE2007, Control Laboratory, University of Twente.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research 26*(5), 433–455.

De Schutter, J., J. Rutgeerts, E. Aertbelien, F. De Groote, T. De Laet, T. Lefebvre, W. Verdonck, and H. Bruyninckx (2005). Unified constraint-based task specification for complex sensor-based robot systems. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 3618–3623. ICRA2005.

Debevec, P. and J. Malik (1997). Recovering high dynamic range radiance maps from photographs. In *Conf. on Computer graphics and Interactive Techniques - SIGGRAPH*, pp. 369–378.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society (Series B) 39*, 1–38.

Devroye, L. (1985). *Non-Uniform Random Variate Generation*. New York: Springer-Verlag.

Dornaika, F. and C. Garcia (1997, July). Robust camera calibration using 2d to 3d feature correspondences. In *International Symposium Optical Science Engineering and Instrumentation - SPIE, Videometrics V*, Volume 3174, pp. 123–133.

Doty, K. L., C. Melchiorri, and C. Bonivento (1993). A theory of generalized inverses applied to robotics. *The International Journal of Robotics Research 12*(1), 1–19.

Etzion, T. (1988). Constructions for perfect maps and pseudorandom arrays. *IEEE Transactions on Information Theory 34*(5/1), 1308–1316.

Fitzgibbon, A. (2001). Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Hawaii, USA. IEEE Computer Society.

Fofi, D., J. Salvi, and E. Mouaddib (2003, July). Uncalibrated reconstruction: an adaptation to structured light vision. *Pattern Recognition 36*(7), 1631–1644.

François, A. (2004). Real-time multi-resolution blob tracking. Technical report, Institute for Robotics and Intelligent Systems, University of Southern California.

Furukawa, R. and H. Kawasaki (2005). Dense 3d reconstruction with an uncalibrated stereo system using coded structured light. In *IEEE Conf. Computer Vision and Pattern Recognition - workshops*, pp. 107–113.

Gao, Y. and H. Radha (2004). A multistage camera self-calibration algorithm. In *IEEE conference on Acoustics, Speech, and Signal Processing*, Volume 3, pp. 537–540.

Ginhoux, R., J. Gangloff, M. de Mathelin, L. Soler, J. Leroy, and J. Marescaux (2003). A 500Hz predictive visual servoing scheme to mechanically filter complex repetitive organ motions in robotized surgery. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, pp. 3361–3366. IROS2003.

Göpel, T., F. Härtl, F. Freyberger, H. Feussner, and M. Buss (2006). Automatisierung eines laparoskopischen nähinstruments. In *Automed*, pp. 1–2.

Griesser, A., N. Cornelis, and L. Van Gool (2006, June). Towards on-line digital doubles. In *Proceedings of the third symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pp. 1002–1009.

Gröger, M., W. Sepp, T. Ortmaier, and G. Hirzinger (2001). Reconstruction of image structure in presence of specular reflections. In *DAGM-Symposium on Pattern Recognition*, Volume 1, pp. 53–60.

Grossberg, M., H. Peri, S. Nayar, and P. Belhumeur (2004). Making one object look like another: controlling appearance using a projector-camera system. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Volume 1, pp. 452–459.

Gudmundsson, S. A., H. Aanæs, and R. Larsen (2007, July). Environmental effects on measurement uncertainties of time-of-flight cameras. In *International Symposium on Signals Circuits and Systems - ISSCS*.

Gühring, J. (2000). Dense 3d surface acquisition by structured light using off-the-shelf components. In *Videometrics and Optical Methods for 3D Shape Measurement*, Volume 4309, pp. 220–231.

Hall-Holt, O. and S. Rusinkiewicz (2001, July). Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Proceedings of the International Conference on Computer Vision*, pp. 359–366.

Hartley, R. and A. Zisserman (2004). *Multiple View Geometry in Computer Vision* (Second ed.). Cambridge University Press, ISBN: 0521540518.

Hayashibe, M. and Y. Nakamura (2001). Laser-pointing endoscope system for intra-operative 3d geometric registration. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 1543–1548. ICRA2001.

Heikkilä, J. (2000, October). Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(10), 1066–1077.

Horaud, R., R. Mohr, F. Dornaika, and B. Boufama (1995). The advantage of mounting a camera onto a robot arm. In *Europe-China workshop on Geometrical Modelling and Invariants for Computer Vision*, Volume 69, pp. 206–213.

Howard, W. (2003). *Representations, Feature Extraction, Matching and Relevance Feedback for Sketch Retrieval*. Ph. D. thesis, Carnegie Mellon University.

Huang, T. and O. Faugeras (1989, December). Some properties of the e matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*(12), 1310–1312.

Inokuchi, S., K. Sato, and M. F. (1984). Range imaging system for 3d object recognition. In *Proc. Int. Conference on Pattern Recognition, IAPR and IEEE*, pp. 806–808.

Isard, M. and A. Blake (1998). Condensation—conditional density propagation for visual tracking. *Int. J. Computer Vision 29*(1), 5–28.

Jonker, P., W. Schmidt, and P. Verbeek (1990, february). A dsp based range sensor using time sequential binary space encoding. In *Proceedings of the Workshop on Parallel Processing*, Bombay, India, pp. 105–115.

Juang, R. and A. Majumder (2007). Photometric self-calibration of a projector-camera system. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, pp. 1–8. IEEE Computer Society.

Kanazawa, Y. and K. Kanatani (1997). Infinity and planarity test for stereo vision. *IEICE Transactions on Information and Systems E80-D*(8), 774–779.

Koninckx, T. (2005). *Adaptive Structured Light*. Ph. D. thesis, KULeuven.

Koninckx, T. P., A. Griesser, and L. Van Gool (2003). Real-time range scanning of deformable surfaces by adaptively coded structured light. In *4th International Conference on 3D Digital Imaging and Modeling*, Banff, Canada, pp. 293–300. IEEE Computer Society.

Koninckx, T. P., P. Peers, P. Dutre, and L. Van Gool (2005). Scene-adapted structured light. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, pp. 611–618. CVPR: IEEE Computer Society.

Konouchine, A. and V. Gaganov (2005). Combined guided tracking and matching with adaptive track initialization. In *Graphicon*, Novosibirsk Akademgorodok.

Kragic, D. and H. Christensen (2001, February). Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation 17*(1), 18–27.

Krupa, A., C. Doigon, J. Gangloff, and M. de Mathelin (2002). Combined image-based and depth visual servoing applied to robotized laparoscopic surgery. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 323–329. IROS2002.

Kyrki, V., D. Kragic, and H. Christensen (2004). New shortest-path approaches to visual servoing. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 1, Sendai, Japan, pp. 349–354. IROS2004.

Lange, R., P. Seitz, A. Biber, and R. Schwarte (1999). Time-of-flight range imaging with a custom solid state image sensor. In *EOS/SPIE International Symposium on Industrial Lasers and Inspection*, Volume 3823, pp. 180–191.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*(2), 150–162.

Li, Y. and R. Lu (2004). Uncalibrated euclidean 3-d reconstruction using an active vision system. *IEEE Transactions on Robotics and Automation 20*, 15–25.

Maas, H. (1992). Robust automatic surface reconstruction with structured light. *International Archives of Photogrammetry and Remote Sensing 29*(B5), 709–713.

Malis, E. and F. Chaumette (2000). 2 1/2d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *International Journal of Computer Vision 37*(1), 79–97.

Malis, E., F. Chaumette, and S. Boudet (1999, April). 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation 15*(2), 234–246.

Malis, E. and R. Chipolla (2000, September). Self-Calibration of zooming cameras observing an unknown planar structure. In *International Conference on Pattern Recognition*, Volume 1, pp. 85–88.

Malis, E. and R. Cipolla (2000, July). Multi-view constraints between collineations: application to self-calibration from unknown planar structures. In *European Conference on Computer Vision*, Volume 2, pp. 610–624.

Malis, E. and R. Cipolla (2002). Camera self-calibration from unknown planar structures enforcing the multi-view constraints between collineations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*(9), 1268–1272.

Marques, C. and P. Magnan (2002). Experimental characterization and simulation of quantum efficiency and optical crosstalk of cmos photodiode aps. In *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications III, SPIE*, Volume 4669.

Mendonca, P. and R. Cipolla (1999). A simple technique for self-calibration. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Volume 1, pp. 500–505.

Mezouar, Y. and F. Chaumette (2002). Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation 18*(4), 534–549.

Morano, R., C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissanov (1998). Structured light using pseudorandom codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*(3), 322–327.

Nayar, S. (1989). Shape from focus. Technical report, Robotics Institute, Carnegie Mellon University.

Nister, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 756–770.

Nister, D., O. Naroditsky, and J. Bergen (2004). Visual odometry. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Volume 1, pp. 652–659.

Nummiaro, K., E. Koller-Meier, and L. Van Gool (2002). A color-based particle filter. In *First International Workshop on Generative-Model-Based Vision*, Volume 1, pp. 53–60.

Oggier, T., B. Büttgen, and F. Lustenberger (2006). Swissranger sr3000 and first experiences based on miniaturized 3d-tof cameras. Technical report, Swiss Center for Electronics and Microtechnology, CSEM.

Orriols, X., A. Willis, X. Binefa, and D. Cooper (2000). Bayesian estimation of axial symmetries from partial data, a generative model approach. Technical Report CVC-49, Computer Vision Center.

Owens, J., D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. Purcell (2007). A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum 26*(1), 80–113.

Pagès, J. (2005). *Assisted visual servoing by means of structured light*. Ph. D. thesis, Universitat de Girona.

Pagès, J., C. Collewet, F. Chaumette, and J. Salvi (2006). A camera-projector system for robot positioning by visual servoing. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 2–9.

Pagès, J., J. Salvi, C. Collewet, and J. Forest (2005). Optimised De Bruijn patterns for one-shot shape acquisition. *Image and Vision Computing 23*(8), 707–720.

Pajdla, T. and V. Hlavác (1998). Camera calibration and euclidean reconstruction from known observer translations. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 421–427.

Perš, J. and S. Kovačič (2002). Nonparametric, model-based radial lens distortion correction using tilted camera assumption. In *Computer Vision Winter Workshop*, pp. 286–295.

Pollefeys, M. (1999). *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences*. Ph. D. thesis, K.U.Leuven.

Pottmann, H., I. Lee, and T. Randrup (1998). Reconstruction of kinematic surfaces from scattered data. In *Symposium on Geodesy for Geotechnical and Structural Engineering*, pp. 483–488.

Pottmann, H., S. Leopoldseder, J. Wallner, and M. Peternell (2002). Recognition and reconstruction of special surfaces from point clouds. *Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV, part 3A, commission III*, 271–276.

Prasad, T., K. Hartmann, W. Weihs, S. Ghobadi, and A. Sluiter (2006). First steps in enhancing 3D vision technique using 2D/3D sensors. In *Computer Vision Winter Workshop - CVWW*, pp. 82–86.

Proesmans, M., L. Van Gool, and A. Oosterlinck (1996). One-shot active 3d shape acquisition. In *International Conference on Pattern Recognition*, Volume III, pp. 336–340.

Qian, G. and R. Chellappa (2004). Bayesian self-calibration of a moving camera. *Comput. Vis. Image Underst. 95*(3), 287–316.

Qian, X. and X. Huang (2004). Reconstruction of surfaces of revolution with partial sampling. *Journal of Computational and Applied Mathematics 163*, 211–217.

Rowe, A., A. Goode, D. Goel, and I. Nourbakhsh (2007). Cmucam3: An open programmable embedded vision sensor. Technical report, Robotics Institute, Carnegie Mellon University.

Rutgeerts, J. (2007). *Task specification and estimation for sensor-based robot tasks in the presence of geometric uncertainty*. Ph. D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

Salvi, J., J. Batlle, and E. Mouaddib (1998, September). A robust-coded pattern projection for dynamic 3d scene measurement. *Pattern Recognition Letters 19*(11), 1055–1065.

Salvi, J., J. Pages, and J. Batlle (2004). Pattern codification strategies in structured light systems. *Pattern Recognition 37*(4), 827–849.

Scharstein, D. and R. Szeliski (2003). High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 195–202.

Schmidt, J., F. Vogt, and H. Niemann (2004, November). Vector quantization based data selection for hand-eye calibration. In *Vision, Modeling, and Visualization*, Stanford, USA, pp. 21–28.

Scholles, M., A. Bräuer, K. Frommhagen, C. Gerwig, H. Lakner, H. Schenk, and M. Schwarzenberger (2007). Ultra compact laser projection systems based on two-dimensional resonant micro scanning mirrors. In *Fraunhofer Publica, SPIE*, Volume 6466.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical J. 27*, 379–423.

Shen, J. and S. Castan (1992). An optimal linear operator for step edge detection. *Computer vision, graphics, and image processing: graphical models and understanding 54, no.2*, 112–133.

Shi, J. and C. Tomasi (1994). Good features to track. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 593–600.

Shirai, Y. and M. Suva (2005). Recognition of polyhedrons with a range finder. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, pp. 71–78. CVPR: IEEE Computer Society.

Soetens, P. (2006, May). *A Software Framework for Real-Time and Distributed Robot and Machine Control*. Ph. D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

Suzuki, S. and K. Abe (1985, April). Topological structural analysis of digital binary images by border following. *Computer Graphics and Image Processing 30*(1), 32–46.

Thormaehlen, T., H. Broszio, and P. Meier (2002). Three-dimensional endoscopy. Technical report, Universität Hannover.

Tomasi, C. (2005). Estimating gaussian mixture densities with em-a tutorial. Technical report, Duke university.

Triggs, B., P. McLauchlan, R. Hartley, and A. Fitzgibbon (2000). Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski (Eds.), *Vision Algorithms: Theory and Practice*, Volume 1883 of *Lecture Notes in Computer Science*, pp. 298–372. Springer-Verlag.

Tsai, R. (1987, August). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation 3*, 323–344.

Tsai, R. (1989). A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation 5*, 345–358.

Vieira, M., L. Velho, A. Sá, and P. Carvalho (2005). A camera-projector system for real-time 3d video. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Volume 3, San Diego, CA, USA, pp. 96–103. CVPR: IEEE Computer Society.

Vincze, M., A. Pichler, and G. Biegelbauer (2002). Automatic robotic spray painting of low volume high variant parts. In *International Symposium on Robotics*.

Vranic, D. and D. Saupe (2001, September). 3d shape descriptor based on 3d fourier transform. In *EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services*, Volume 1, pp. 271–274.

Vuylsteke, P. and A. Oosterlinck (1990, February). Range image acquisition with a single binary-encoded light pattern. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*(2), 148–164.

Xu, D., L. Wang, Z. Tu, and M. Tan (2005). Hybrid visual servoing control for robotic arc welding based on structured light vision. *Acta Automatica Sinica 31*(4), 596–605.

Zhang, B., Y. Li, and Y. Wu (2007). Self-recalibration of a structured light system via plane-based homography. *Pattern Recognition 40*(4), 1368–1377.

Zhang, D. and G. Lu (2002). A comparative study of fourier descriptors for shape representation and retrieval. In *Asian Conference on Computer Vision*, pp. 646–651.

Zhang, L., B. Curless, and S. Seitz (2002, June). Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 24–36.

Zhang, R., P. Tsai, J. Cryer, and M. Shah (1994). Analysis of shape from shading techniques. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 377–384.

Zhang, Y., B. Orlic, P. Visser, and J. Broeninck (2005, November). Hard real-time networking on firewire. In *7th real-time Linux workshop*.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(11), 1330–1334.

Appendix A

Pattern generation algorithm

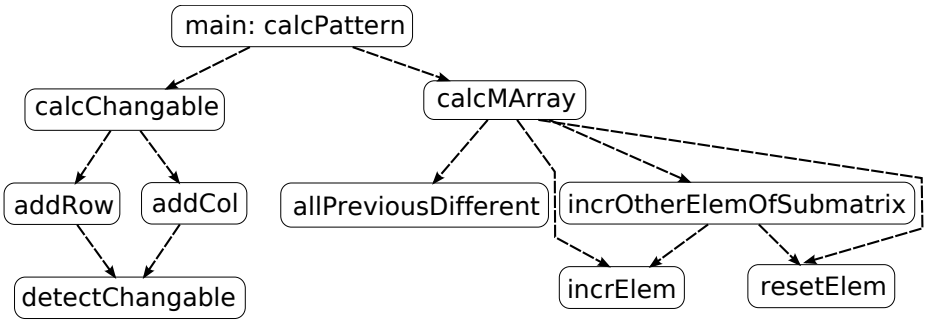


FIGURE A.1: Overview of dependencies in the algorithm methods

Algorithm A.1 Main: calculation of every larger suitable patterns: calcPattern($maxSize$, $aspectRatio$, $minH$)

for $i \leftarrow 1$ to $maxSize$ **do**
  for $j \leftarrow 1$ to $aspectRatio * maxSize$ **do**
    $MArray_{i,j} \leftarrow 0$
  end for
end for
$processing \leftarrow 1$
calcChangable()
$noCols \leftarrow w$
while true do
  calcMArray($noCols$, $processing$, $minH$)
  $noCols \leftarrow noCols + 1$
end while

Algorithm A.2 calcChangable()

index ← 1
for $c \leftarrow 1$ to $maxSize - w$ **do**
  addCol(r, c, *index*)
  $r_{prev} \leftarrow r$
  $r \leftarrow \text{round}(\frac{3c}{4})$
  if $r > r_{prev}$ **then**
    addRow(r, c, *index*)
  end if
end for

Algorithm A.3 addCol(r, c): add changable info for c^{th} column up to r rows (analogous for dual function: addRow)

for $i \leftarrow 1$ to r **do**
  begin, *end* ← detectChangable(i, c)
  markAsRead(i, c)
  $changable_{index} \leftarrow (i, c, begin, end)$
  $index \leftarrow index + 1$
end for

Algorithm A.4 *begin*, *end* ← detectChangable(i, j): (i, j) is the upper left element of the submatrix

{Sequence of reading the elements in a submatrix: $\begin{smallmatrix}0&1&2\\7&8&3\\6&5&4\end{smallmatrix}$ }
if markedAsRead(i, $j + 2$) **then**
  $begin \leftarrow 2\{(0, 2)\}$
else
  $begin \leftarrow 4\{(2, 2)\}$
end if
if markedAsRead($i + 2$, j) **then**
  $end \leftarrow 6\{(2, 0)\}$
else
  $end \leftarrow 4\{(2, 2)\}$
end if

Algorithm A.5 calcMArray($noCols$, $processing$, $minH$); resetElem($index$): set the blobs in the $index^{th}$ processing step to 0

```
toProcess ← getProcessIndexAtColumn(noCols)
while processing < toProcess do
  allDiff, conflictIndex ← allPreviousDifferent(processing, minH)
  if allDiff = true then
    processing ← processing + 1
  else
    incrPossible ← incrElem(processing)
    if NOT incrPossible then
      incrPossible ← increaseOtherElemOfSubmatrix(processing)
    end if
    if NOT incrPossible then
      resetAllPreviousProcessingStepsUpTo(conflictIndex)
      processing ← conflictIndex
      incrPossible ← incrElem(processing)
    end if
    if NOT incrPossible then
      incrPossible ← increaseOtherElemOfSubmatrix(processing)
    end if
    while NOT incrPossible do
      resetElem(processing)
      if processing > 0 then
        processing ← processing − 1
        incrPossible ← incrElem(processing)
      else
        Search space exhausted: pattern impossible.
      end if
    end while
  end if
end while
```

Algorithm A.6 increaseOtherElemOfSubmatrix($processing$)

```
incrPossible ← false
otherElemsFound ← 0
while (otherElemsFound + elemsInThisSubmatrix() < w^2) AND NOT
incrPossible do
  repeat
    resetElem(processing)
    processing ← processing − 1
  until elemsPartOfSubmatrix(processing) OR processing = 0
  otherElemsFound ← otherElemsFound + 1
  incrPossible ← incrElem(processing)
end while
return incrPossible
```

Algorithm A.7 allPreviousDifferent($lastPos$, $minH$): return **true** if all changable elements up to index lastPos are different from the last one

$allDiff \leftarrow$ **true**
$i \leftarrow 1$
$last \leftarrow$ getSubmatrix($endPos$)
while $allDiff$ AND ($i < endPos$) **do**
  $hamming_j \leftarrow$ compareSubmatrix($last$, getSubmatrix(i))
  for $j \leftarrow 1$ to 4 **do**
    $allDiff \leftarrow allDiff$ AND ($hamming_j \geq minH$)
  end for
  $i \leftarrow i + 1$
end while
$i \leftarrow i - 1$
return $allDiff$, i

Algorithm A.8 compareSubmatrix($subMat1$, $subMat2$): returns the hamming distance between submatrices for every rotation

for $i \leftarrow 1$ to 4 **do**
  $hamming_i \leftarrow$ |sgn(centralBlob($subMat1$) - centralBlob($subMat2$))|
  for $j \leftarrow 1$ to $w^2 - 1$ **do**
    $hamming_i \leftarrow hamming_i$ + |sgn(otherBlob($subMat1$, 1 + ((i+2*j-1)mod $w^2 - 1$)) - otherBlob($subMat2$, j))|
  end for
end for

Algorithm A.9 incrElem($index$): increase the value of the blobs in the $index^{th}$ processing step

$string \leftarrow$ getChangableElemsOfThisSubmatrix($changable_{index}$)
$state \leftarrow$ baseAtobase10($string$)
if $state < a^{changable.stop - changable.start + 1} - 1$ **then**
  $state \leftarrow state + 1$
  $string \leftarrow$ base10tobaseA($state$)
  setChangableElemsOfThisSubmatrix($changable_{index}$, $string$)
  return true
else
  return false
end if

Appendix B

Labelling algorithm

Algorithm B.1 find4Closest(B)

for $i \leftarrow 0$ to $|B| - 1$ **do**

$S \leftarrow S \subset B : S = \{B_k\}$:max($|u_{B_i} - u_{B_k}|$, $|v_{B_i} - v_{B_k}|$) $< 3 * \sqrt{\frac{\mathcal{W} * \mathcal{H}}{|B|}}$, $0 \leq k \leq |B| - 1$

$N_{i,0} \leftarrow N_{i,0} \in S : \arg\min_k \parallel (u_{N_{i,0}} - u_{S_k}, v_{N_{i,0}} - v_{S_k}) \parallel_2, 0 \leq k \leq |S| - 1$

$\theta_{i,0} \leftarrow \arctan \frac{v_{N_{i,0}} - v_{B_i}}{u_{N_{i,0}} - u_{B_i}}$

$N_{i,2} \leftarrow 2^{nd}$ closest k with $|\theta_{i,2} - \theta_{i,0}| > \frac{\pi}{3}$: $\frac{(u_k - u_i)(u_j - u_i) + (v_k - v_i)(v_j - v_i)}{\parallel (u_k - u_i, v_k - v_i) \parallel_2 \parallel (u_j - u_i, v_j - v_i) \parallel_2} < 0,5$

$N_{i,4}, N_{i,6} \leftarrow 3^{rd}\&4^{th}$ closest k with $\theta_{i,4}, \theta_{i,6}$ such that $> \frac{\pi}{3}$ of all other N_i

$d_{avg} \leftarrow \frac{\sum_{j=1}^{4} \parallel (u_j - u_i, v_j - v_i) \parallel_2}{4}$

$\mathbf{u}_{tot} \leftarrow \sum_{j=1}^{4} (u_j - u_i, v_j - v_i)$

if $\parallel \mathbf{u}_{tot} \parallel_2 > d_{avg}$ **then**

$B \leftarrow B \setminus B_i$

end if

end for

Algorithm B.2 testGraphConsistency()

for $i \leftarrow 0$ to $|B| - 1$ **do**
  $consist_{center} \leftarrow$ **true**
  for $j \leftarrow 0$ to 7 **do**
    $\theta \leftarrow \theta_{i,j} + \pi$
    if $\theta > 2\pi$ **then**
      $\theta \leftarrow \theta - 2\pi$
    end if
    $O_j \leftarrow j + 4 \bmod 8$
    while $|(\theta_{i,O_j} \circ N_{i,j}) - \theta| > \frac{\pi}{16}$ **do**
      $O_j \leftarrow O_j + 1 \bmod 8$
    end while
    $consist_{center} \leftarrow consist_{center}\ AND\ (B_i = N_{i,O_j} \circ N_{i,j})$
  end for
  for $j \leftarrow 0$ to 3 **do**
    $consist_{2j+1} \leftarrow \{N_{i,(O_{2j}-2) \bmod 8} \circ N_{i,2j} = N_{i,2j+1} = N_{i,(O_{(2j+2) \bmod 8}+2) \bmod 8} \circ N_{i,(2j+2) \bmod 8}\}$
    $consist_{2j} \leftarrow \{N_{i,(O_{(2j-1) \bmod 8}-1) \bmod 8} \circ N_{i,(2j-1) \bmod 8} = N_{i,2j} = N_{i,(O_{(2j+2) \bmod 8}+1) \bmod 8} \circ N_{i,2j+2 \bmod 8}\}$
  end for
  $consist \leftarrow consist_{center}$
  for $j \leftarrow 0$ to 7 **do**
    $consist \leftarrow consist\ AND\ consist_j$
  end for
  if $NOT\ consist$ **then**
    $B \leftarrow B \setminus \{B_i, \text{involved } N_{i,k}\}$
  end if
end for

Algorithm B.3 findCorrespondences(), assuming $h = 3$

for $i \leftarrow 0$ to $|B| - 1$ **do**
  $pos \leftarrow$ codeLUT(i)
  if *pos* invalid **then**
    $doubt_i \leftarrow$ **true**
    for $j \leftarrow 0$ to 8 **do**
      while *pos* invalid AND more letters available **do**
        increment (copy of) code in $N_{i,j}$ or B_i when $j = 8$
        $pos \leftarrow$ codeLUT(i)
      end while
      if *pos* valid **then**
        $votes_{B_i,code} \leftarrow votes_{B_i,code} + 1$
        for $k \leftarrow 0$ to 7 **do**
          $votes_{N_{i,k},code} \leftarrow votes_{N_{i,k},code} + 1$
        end for
      end if
    end for
  else
    $doubt_i \leftarrow$ **false**
    $votes_{B_i,code} \leftarrow votes_{B_i,code} + 9$
    for $k \leftarrow 0$ to 7 **do**
      $votes_{N_{i,j},code} \leftarrow votes_{N_{i,j},code} + 9$
    end for
  end if
end for
for $i \leftarrow 0$ to $|B| - 1$ **do**
  if $doubt_i =$ **true then**
    code of $B_i \leftarrow \arg\max_{code} votes_{B_i,code}$
  end if
end for
for $i \leftarrow 0$ to $|B| - 1$ **do**
  $pos \leftarrow$ codeLUT(i)
  if *pos* valid **then**
    convertPosToProjectorCoordinates()
  end if
end for

Algorithm B.4 codePos ← codeLUT(blobPos)

$s_i \leftarrow$ string of length 9 base a: B_i, Ni, j, $0 \leq j \leq 7$
$dec \leftarrow convertBase(a, 10, s_i)$
$pos \leftarrow$ binarySearch(dec, preProcessingList)

Appendix C

Geometric anomaly algorithms

C.1 Rotational axis reconstruction algorithm

First, θ and ϕ are calculated, and then, at the very end of this paragraph, the 2D point in the XY plane is calculated.

In a first step a quality number for each selected (θ,ϕ) pair is computed, and the best of those orientations is selected. Since this is the part of the algorithm that is most often evaluated, it is likely to be the bottleneck on the throughput. Therefore an estimation of computational cost is appropriate here.

Let V be the number of vertices of the mesh, and T be the number of triangles in the mesh.

For each (θ,ϕ) pair to be tested:

1. Consider the plane through the origin perpendicular to the direction chosen, which can be written using spherical coordinates as:

$$\cos(\theta)\sin(\phi)x + \sin(\theta)\sin(\phi)y + \cos(\phi)z = 0$$

   Compute the distances d_j from the vertices j to that plane:

$$d_j = \cos(\theta)\sin(\phi)x_j + \sin(\theta)\sin(\phi)y_j + \cos(\phi)z_j$$

   This step (computing d_j) requires $5V$ flops and four trigonometrical evaluations. The aim is to test the intersection of these planes with the mesh for its circularity, as is explained later on.

2. Construct P+1 parallel planes parallel to that plane, for P a small number, e.g. $P = 10$ (see fig. 8.7):

$$\cos(\theta)\sin(\phi)x + \sin(\theta)\sin(\phi)y + \cos(\phi)z - D_i = 0 \text{ for } i = 0 \dots P$$

   Choose D_i such that the planes are spread uniformly in the region where they intersect with the mesh: let

$$\Delta D = \frac{max_j(d_j) - min_j(d_j)}{P} \Rightarrow D_0 = min_j(d_j) \text{ and } D_i = D_{i-1} + \Delta D$$

for $i = 1 \ldots P$ This requires the calculation of two extrema over V.

3. We will now determine in between which two consecutive planes each vertex falls, calling the space between plane i and $i+1$ layer i. For each vertex j, calculate
$$layer_j = \lfloor \frac{d_j - min_i(d_i)}{\Delta D} \rfloor$$
Hence, vertex j is in between the plane with index $i = layer_j$ and $i = layer_j + 1$. This requires $2V$ flops, V floor evaluations. Note that there are now $P-1$ intersections between the mesh and the planes to be checked for circularity, since the plane with index $i = 0$ and the plane with index $i = P$ intersect the mesh in only one vertex: the one with the minimum d_j for the former and the one with the maximum d_j for the latter.

4. For each triangle in the mesh: Let a equal the average of the layer numbers of the three corners of the triangle. If the result is not an integer (and thus not all the layer numbers of the three corners are the same), then the triangle is intersected by the plane indexed $\lfloor a \rfloor + 1$. In that case, rotate the triangle over $-\theta$ around the z axis and $-\phi$ around the y axis (rotate (θ, ϕ) back to the z axis). Triangles are considered small, hence omit calculating the intersection of the plane with the triangle but use the average of the corner values instead. Construct a bag of 2D intersection points for each plane. Then add the x and y coordinate of the average corner coordinates to the bag corresponding to the intersected plane. This costs 3T flops, T tests which if successful each take 8*3+6 flops more. Hence, $\approx (3T + 30P\sqrt{T})$ flops and T tests.

5. For each of the bags fit a circle through its intersection points. A non-iterative circle fitting algorithm is used as described in [Chernov and Lesort, 2003], minimising
$$F(a,b,R) = \sum_{k=1}^{n} [(x_i - a)^2 + (y_i - b)^2 - R^2]^2 = \sum_{k=1}^{n} [z_i + Bx_i + Cy_i + D]^2$$
with $z_i = x_i^2 + y_i^2, B = -2a, C = -2b, D = a^2 + b^2 - R^2$. Differentiating F with respect to B,C and D yields a linear system of equations, only using $13n + 31$ flops with n the number of 2D points.
The initial mesh is a noisy estimate, we deal with that noise using a RANSAC scheme [Hartley and Zisserman, 2004] over the intersection points to eliminate the outliers in the set of intersection points. RANSAC can be used here because most of the points will have little noise, and only a small fraction of them is very noisy (like the burr). The following algorithm determines the number of iterations N needed. Let s be the number of points used in each iteration, using a minimum yields $s = 3$ (a circle has three parameters). Let ϵ be the probability that the selected data point is an outlier. If we want to ensure that at least one of the random samples

of s points is free from outliers with a probability p, then at least N selections of s points have to be made with $(1-(1-\epsilon)^s)^N = 1-p$. Hence, the algorithm for determining N:

$N = \infty, i = 0$, `while` $N > i$

- Randomly pick a sample (three points) and count the number of outliers
- $\epsilon = \dfrac{number\ of\ rejectedpoints}{number\ of\ points}$
- $N = \dfrac{log(1-p)}{log(1-(1-\epsilon)^s)}$
- increment i

Applying this to the wheel data yields:

tolerance$[mm]$	2.5	1.7	.83	.42	.21
% rejected	6	8	16	46	61
$\lceil N \rceil$	3	4	6	28	73

As the noise level is in the order of magnitude of $1mm$, 5 RANSAC iterations will do. In each iteration, take 3 points at random and fit a circle through these points using the algorithm described. Determine how many of all the points lie within a region of $1mm$ on each side around the estimated circle. After the iterations, use the circle estimate corresponding to the iteration that had the most points inside the region to remove all points outside this uncertainty interval.
Since the number of iterations and the fraction of the data used is small, the computational cost of the iterations can be neglected. As can be seen in figure 8.3 the data is roughly outlined on a rectangular grid. Since the triangles in the mesh have about equal size, the number of triangles intersected by a plane is in the order of magnitude of the square root of the total number of triangles. Hence, removing the points outside the tolerance interval costs $\approx P(8\sqrt{T})$ flops and $\approx P\sqrt{T}$ tests.
Afterwards run the circle fitter algorithm on all points but the removed outliers: this step costs $\approx P(13\sqrt{T}+34)$ flops, and two extrema over P which can be neglected since $P \ll V$.

6. To measure the quality of this orientation: after estimating the circle, compute the distances between each intersection point and that circle. Average over all the intersection points in a circle, and over all circles. Return the result as the quality number. See section 8.3.5 for results.

Approximating cost of computing the two extrema over V and the floor over V as $3V$ flops, this brings the cost of the algorithm on $\approx 10V+3T+P(51\sqrt{T}+34)$ flops and $2(V-1)+T$ tests.
For every triangulation $T = V_o + 2V_i - 2$ with V_o the vertices at the edges of the mesh, and V_i the vertices inside the mesh ($V = V_i + V_o$). Starting from empty data structures, the first triangle is only constructed at the third point, hence

"-2". Every point that is added outside this triangle adds a new triangle, hence "V_o". Every point that is added inside one of the triangles divides that triangle in three, or two triangles are added, hence the "$2V_i$".
This however is only valid for meshes that consist of only one triangulation strip. Otherwise, the formula is only valid for each strip. In our case, most meshes are built up as a single strip.
Again approximating the mesh as a square deformed in three dimensions, the number of vertices in the mesh is about the number of vertices along one of the sides squared. Therefore, approximating V_o as $4\sqrt{V}$ and V_i as $V - V_o$, the cost becomes

$$\approx 16V - 12\sqrt{V} + 51\sqrt{2}P\sqrt{V - 2\sqrt{V)}} \text{ flops and } 4(V - \sqrt{V}) \text{ tests}$$

hence O(V). For big meshes $\approx 16V$ flops and $4V$ test.

C.2 Burr extraction algorithm

In the case studied—a wheel—the geometrical anomaly is parallel to the axis orientation. It is assumed to be in this outline of the algorithm:

- For each of the $P-1$ circular intersections of the winning axis orientation, determine the distance between each intersection point and the estimated circle (no extra calculations needed: this has been done in section 8.3.3).
- Then find the location on each circle of the point (bx_i, by_i) with the maximum distance ($i = 1...P-1$). For each circle that location is defined by the angle α_i in the circle plane with the circle centre (cx_i, cy_i) as origin of the angle. In section 8.3.3 the intersected points have been rotated from orientation (θ, ϕ) back to the Z axis. That data can now be reused to calculate the angles: the Z coordinate can simply be dropped to convert the 3D data into 2D data: `for` $i = 1...P-1 : \alpha_i = tan^{-1}(\frac{by_i - cy_i}{bx_i - cx_i})$
- The lines in fig.8.10 all indicate the burr orientation correctly, such that the average of the angles α_i would be a good estimate of the overall burr angle α. For figure 8.11 for example: $\alpha = \frac{\alpha_1 + \alpha_2 + ... + \alpha_{P-3} + \alpha_{P-1}}{P-1}$. However, the burr may have been too faint to scan on some places on the surface. Hence it is possible that some circular intersections do not have the correct α_i. Therefore, to make it more robust, a RANSAC scheme is used over those $P-1$ angles, with $s = 1$. Assuming no more than a quarter of the angles are wrong and requiring a 99% chance that at least one suitable sample is chosen, the number of iterations $N = \lceil \frac{log(0.01)}{log(.25)} \rceil = 4$. Choose the tolerance e.g. $t = 5\pi/180$. For N iterations: randomly select one of the circles i_{Rand} and determine how many of the other circles have their angle within an angle t of the burr angle of this circle: $|\alpha_{i_{Rand}} - \alpha_i| < t$.
- Select the circle where the tolerance test was successful most often, discard circles that have a α_i outside the tolerance t. Then average the α_i over the remaining circles (see fig. 8.11), this is the burr angle α.

Index

3D acquisition, 12

active sensing, 59, 193
alphabet, 27, 43
aspect ratio, 30, 82
assumptions, 100, 102

background segmentation, 100
barrel distortion, 74
baseline, 75, 79, 80
Bayer, 135, 156
Bayes' rule, 108
Bayesian filtering, 190, 193
blooming, 47

calibration, 25
    - object, 80
    geometric -, 75
    hand-eye -, 91, 95, 97, 164
    intensity -, 62
    projector -, 72
    self-, 81
CCD, 47, 134, 135
central difference, 65
chromatic crosstalk, *see* crosstalk, chromatic
clustering, 194
CMOS, 47, 134
collineation, 89, 95
    super-, 93
colour space
    HSV, 44, 161
    Lab, 44
    RGB, 44
coloured projection pattern, *see* projection pattern, coloured
communication channel, 42
condensation algorithm, 109
conditioning, 26, 122
constraint based task specification, 139, 145
correspondence problem, 15, 76
crosstalk
    chromatic -, 62, 167
    optical -, 47
cyclic permutation, 29, 114

De Bruijn, 28
decomposition
    QR -, 92
discontinuity of the scene, 57
DLP, 20, 179
DMD, 20

eigenvalue problem, 119
EM segmentation, 103, 187
entropy, 41
epipolar geometry, 85, 96
error correction, 30
Euler angles, 77, 82, 121, 137, 143
exponential map, 82
extrinsic parameters, 25, 77, 81
eye-in-hand, eye-to-hand, 20

feature tracking, *see* tracking, feature
FFT, 49, 53
finite state machine, 154
FireWire, 134
floodfill segmentation, 101
focal length, 69
focus, out of-, 58

focus, shape from..., 16
FPGA, 157

Gauss-Jordan elimination, 120
Gaussian mixture model, 103, 152
GPU, 157
gradient descent, 171
graph consistency, 115
grey scale projection pattern, *see* projection pattern, grey scale

Hamming distance, 25, 30, 162, 192
hexagonal maps, 37
histogram, 103
homography, 89, 96
    super-, 93
HSV, *see* colour space, HSV

IEEE1394, 134, 148, 153
IIDC, 134, 148
image based visual servoing, 20
information theory, 18
intensity calibration, *see* calibration, intensity
interferometry, shape from..., 16
intrinsic parameters, 25, 69, 81
inverse kinematics, 145
inversion sampling, 187
ISEF filter, 186

Jacobian, 83
    image -, 10
    robot -, 10, 137

Kalman filtering, 109

Lab, *see* colour space, Lab
labelling, 114
Lambertian reflection, 62, 162
laser, 189, 193, 194
LCD, 20, 179
LDA, 194
LED projector, 22, 193, 194
lens aberration, 82
Levenberg-Marquardt, 171

MAP estimation, 108
matrix
    - of dots, 29
    essential, 86
    fundamental, 86
Monte Carlo, 47
movement, scene -, 58
multi-slit pattern, 28, 43

object recognition, 194
optical crosstalk, *see* crosstalk, optical
optimisation, 81
    non-linear, 80, 83
oversaturation, 64

P-controller, 144
P-tile segmentation, 105
particle filtering, 84, 96, 109
PCA, 194
PDF, 103, 106
perfect map, 29
pincushion distortion, 74
pinhole model, 19, 69, 72, 83, 110, 117
planarity, 88, 192
principal distance, 69
principal point, 70, 82, 143
prior PDF, 106
projection pattern
    adaptation of-, 59
    brightness of-, 101
    coloured, 44
    coloured -, 189
    grey scale, 47
    shape based, 51
    spatial frequencies, 52
pseudoinverse, 118

quaternions, 77, 91

radial distortion, 73, 90, 115
RANSAC, 86, 172, 214
reconstruction
    equations, 78
    uncalibrated, 79

reflection models, 57, 62, 190
RGB, 28, *see* colour space, RGB
rotational invariance, 33

segmentation, 100
self occlusion, 110, 192
sensor integration, 192, 193
sensors, 1
shading, shape from. . . , 16
shape based projection pattern, *see* projection pattern, shape based
shape from X, *see* X, shape from . . .
silhouettes, shape from . . . , 16
singularity, 87
sinusoidal intensity variation, 52
skew, 82, 87
SLAM, 14
spatial encoding, 26
spatial frequencies projection pattern, *see* projection pattern, spatial frequencies
specular reflection, 62, 146, 167
stereo, 13, 20
stripe pattern, 28
structure from motion, 84
structured light, 4
subspace methods, 194
SVD decomposition, 65, 119

texture, shape from. . . , 16
textured scene, 192
time multiplexing, 25
time of flight, 12, 193
tracking
    3D -, 193
    calibration -, 97
    feature -, 109
triangulation, 13, 75, 117

UDP, 135
UML, 148

VGA, 42, 135
vignetting effect, 68
virtual parallax, 89, 96
visual servoing
    image based -, 9, 79
    position based, 10
voting, 187

Resume



Personal data

Kasper Claes
January 4 1979, Berchem, Belgium
kasper.claes@gmail.com
http://people.mech.kuleuven.be/∼kclaes

Education

- **2004 - 2008**: **Ph.D. in mechanical engineering** at the Katholieke Universiteit Leuven, Belgium.

  My research is situated in the area of 6DOF robot control using structured light. 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.

- **2001 - 2002**: **Master in social and cultural anthropology** at the Katholieke Universiteit Leuven, Belgium.

- **1998 - 2001**: **Master of science in computer science** specialisation mechatronics at the Katholieke Universiteit Leuven, Belgium.

  2000 - 2001: **EPFL** Master thesis at the EPFL in Lausanne, Switzerland (stay of one semester).

  2001: **Athens** program at the Ecole Nationale Supérieure des Techniques Avancées, Paris, France.

- **1996 - 1998**: **Bachelor in engineering** at the Katholieke Universiteit Leuven, Belgium.

List of publications

R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter. iTasc: a tool for multi-sensor integration in robot manipulation. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul, South-Korea, 2008. MFI2008.

K. Claes and H. Bruyninckx. Robot positioning using structured light patterns suitable for self calibration and 3d tracking. In *Proceedings of the International Conference on Advanced Robotics*, pages 188–193, August 2007.

J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

K. Claes and H. Bruyninckx. Endostitch automation using 2d and 3d vision. Internal report 06PP160, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, 2006.

K. Claes, T. Koninckx, and H. Bruyninckx. Automatic burr detection on surfaces of revolution based on adaptive 3D scanning. In *5th International Conference on 3D Digital Imaging and Modeling*, pages 212–219. IEEE Computer Society, 2005.

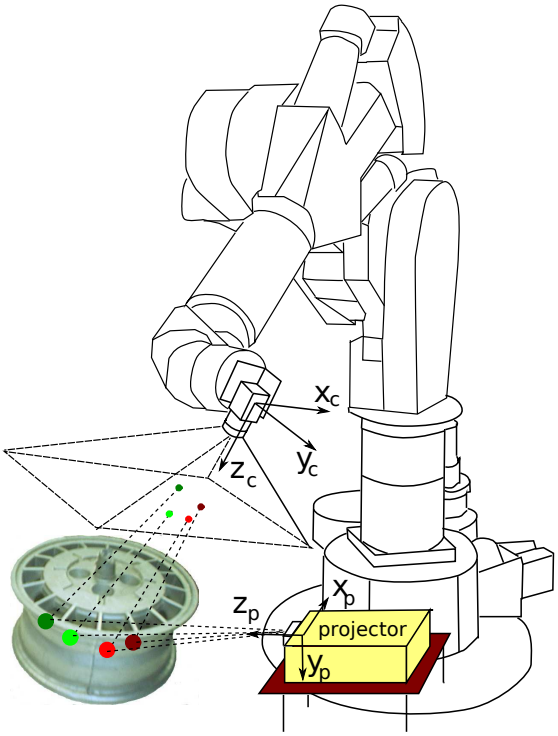
K. Claes and G. Zoia. Optimization of a virtual dsp architecture for mpeg-4 structured audio. *Laboratoire de traitement de signaux, EPFL, Lausanne, Switzerland*, pages 1–57, 2001.

Gestructureerd licht aangepast aan de controle van een robotarm

Nederlandstalige samenvatting

1 Inleiding

De meeste industriële robotarmen maken alleen gebruik van proprioceptieve sensoren: die bepalen welke de posities zijn van de verschillende gewrichten van de robot. Deze thesis kadert in het gebruik van exteroceptieve sensoren voor de controle van een robotarm. Exteroceptieve sensoren zijn sensoren waarmee de buitenwereld waargenomen wordt. Meer bepaald gaat het over één exteroceptieve sensor: een camera. De bedoeling is om de afstand tot de verschillende elementen in de scene in te schatten, en daarvoor zijn herkenbare visuele elementen in het beeld nodig. Als die er niet zijn, kan een projector soelaas bieden. De projector vervangt dan een tweede camera, en het geprojecteerde licht zorgt voor de nodige visuele elementen die er anders niet zouden zijn. Figuur 1 laat de opstelling zien die het meest bestudeerd wordt doorheen deze thesis.
De resulterende 3D reconstructie is geen doel op zich, maar een middel om concrete robottaken tot een goed einde te brengen. Die resolutie van die reconstructie is niet hoger dan nodig is voor de robottaak: het gros van de data van de gebruikelijke fijne 3D-reconstructie zou toch maar een verspilling zijn van rekenkracht.
De typische toepassingen zitten in de industriële en medische wereld. Bijvoorbeeld bij het verven van industriële onderdelen die egaal van kleur zijn. Ook menselijke organen hebben erg weinig natuurlijke beeldelementen. In dit soort van gevallen is het gebruik van gestructureerd licht nuttig.
Hoewel deze thesis enkel deze ene sensor bestudeert, is het voor het tot een goed einde brengen van een robottaak van belang om de informatie van meerdere sensoren te integreren, zoals ook wij als mens voortdurend gebruik maken van meerder zintuigen.



Figuur 1: De opstelling die doorheen de thesis bestudeerd wordt

1.1 Open problemen en bijdragen

Zelfs na meer dan een kwart eeuw onderzoek naar gestructureerd licht [Shirai and Suva, 2005], blijven bepaalde problemen open :

- Probleem: Vaak blijft de pose tussen camera en projector constant. Die opstelling zorgt voor eenvoudige wiskunde om de afstanden in te schatten. Maar het is interessant om de camera te laten meebewegen met de eindeffector van de robot: de visuele gegevens worden meer of minder gedetailleerd naargelang de beweging. De huidige generatie projectoren laten technisch niet toe om mee te bewegen. Pagès et al. [2006] werkte ook al met deze veranderende relatieve positie, maar hij gebruikt niet het volledige wiskundige potentieel ervan: de geometrische calibratie daar gebeurt tussen verschillende cameraposities en niet tussen camera en projector.
  Bijdrage: Zorgen voor een calibratie tussen camera en projector, wat de triangulatie robuuster maakt dan tussen verschillende cameraposities. Die geometrische parameters worden tijdens de beweging aangepast.

- Probleem: Normaalgezien hebben de camera en de projector vergelijkbare oriëntaties: wat boven, onder, links en rechts is in het projectorbeeld blijft

zo in het camerabeeld. In deze opstelling kan de camera niet alleen in drie richtingen transleren, maar ook in drie richtingen roteren ten opzichte van de projector. Salvi et al. [2004] geeft een overzicht van het onderzoek over gestructureerd licht over de laatste decennia: elk van die technieken steunt op een gekende rotatie tussen camera en projector, meestal quasi geen rotatie.
Bijdrage: Nieuw in dit werk is de onafhankelijkheid van de patronen van de relatieve rotatie tussen camera en projector.

- Probleem: Voor robotica zijn tweedimensionale patronen nuttig, die op basis van één beeld een reconstructie kunnen maken. Op die manier is een willekeurige oriëntatie tussen camera en projector toegelaten, en mag de scene bewegen. De bestaande methoden daarin functioneren op basis van kleuren in het patroon [Adán et al., 2004, Chen et al., 2007], maar die methoden falen op gekleurde scenes, omdat dan delen van het patroon niet gereflecteerd worden. De enige oplossing daarvoor is het aanpassen van de kleuren van het patroon aan de scene, maar dan is heeft de reconstructie verschillende videobeelden nodig, en dat legt beperkingen op aan de beweeglijkheid van de scene.
  Bijdrage: De techniek die we voorstellen hangt niet af van kleuren: het is gebaseerd op de relatieve verschillen in intensiteitswaarden. Het is een techniek op basis van één beeld, onafhankelijk van de kleur van de scene.

- Probleem: Gestructureerd licht houdt een weging in tussen robuustheid en 3D-resolutie: hoe fijner die resolutie, hoe groter de kans om geprojecteerde elementen verkeerd te decoderen. In dit werk gaat het niet om een precieze reconstructie, dan wel om het ruwere interpreteren van welke voorwerpen zich waar in de scene bevinden. Tijdens de robotbeweging verzamelt de robot gradueel meer informatie over de scene: de informatieinhoud in elk van de opgenomen beelden hoeft niet overdreven hoog te zijn.
  Bijdrage: Deze thesis kiest voor een lage 3D-resolutie en een hoge robuustheid. We veranderen de grootte van de elementen in het projectorbeeld om de 3D-resolutie online aan te passen naargelang de noden in de robottaak op dat moment.

- Probleem: Vaak zorgen diepteverschillen voor occlusies van delen van het geprojecteerde patroon. Zonder foutcorrectie, kunnen die delen niet gereconstrueerd worden, omdat het nodig is de naburige lichtpunten te herkennen om de associatie te kunnen maken tussen punten in camera- en projectorbeeld.
  Bijdrage: Omdat de resolutie in het projectorbeeld laag is, kunnen we ons permitteren om de redundantie erin te verhogen. Dit verhoogt ook de robuustheid, maar op een geheel andere manier dan in de vorige bijdrage. We voegen foutcorrigerende codes toe in het beeld. De code is zo dat er niet meer intensiteitniveau's nodig zijn dan bij andere technieken, maar dat wanneer één van de elementen niet zichtbaar is, die kan gecorrigeerd

worden. Of, met andere woorden, voor een constant aantal te corrigeren fouten, is de resolutie van het projectorbeeld met onze techniek groter.

- Probleem: In voorgaand werk wordt nauwelijks expliciet het verschil vermeld tussen de code in een patroon, en de manier waarop die code geprojecteerd wordt. Het resultaat is dat sommige van de mogelijke combinaties niet bestudeerd worden.
  Bijdrage: Deze thesis scheidt de methoden die de logica van abstracte patronen genereert, en de manier waarop die in praktijk worden gezet. Het bestudeerd uitgebreid de verschillende mogelijkheden om de patronen in de praktijk te zetten, en maakt telkens meest geschikte keuze voor robotica expliciet.

- Probleem: Hoe kan nu de resulterende puntenwolk gebruikt worden om een robottaak uit te voeren?
  Bijdrage: We passen de technieken van beperkingsgebaseerde taakspecificatie toe bij dit gestructureerd licht: dit zorgt voor een wiskundig elegante manier om een taak te specifiëren op basis van 3D-informatie, en laat een eenvoudige integratie toe met data die komt van andere sensoren, mogelijk op heel andere frequenties.

- Probleem: Hoe zeker zijn we van de metingen die het gestructureerde lichtsysteem aanlevert, hoe vertalen die zich naar geometrische toleranties voor de robottaak?
  Bijdrage: Deze thesis presenteert een evaluatie van de mechanische fouten, gebaseerd op een willekeurige positie van projector en camera in de 6D ruimte. Dit is een hoogdimensionale foutenfunctie, maar door bepaalde goed gekozen veronderstellingen te maken, wordt het duidelijk welke variabelen gevoelig zijn voor fouten in welk gebied.

1.2 3D-sensoren

Er zijn verschillende manieren om een robotarm visueel te controleren. De twee belangrijkste zijn beeld- en positiegebaseerde controle. De eerste stuurt de robot rechtstreeks bij op basis van de coordinaten in het beeld, een andere reconstrueert eerst de diepte en stuurt de robot op basis daarvan bij. Beiden hebben voor- en nadelen. Er zijn ook hybride vormen die de beiden combineren. Wat ze gemeenschappelijk hebben is dat voor elke vorm van controle van een robotarm op basis van videobeelden, diepteinformatie nodig is. Die informatie kan op verschillende manieren verkregen worden:

- Vluchttijd: de diepte wordt gemeten op basis van het tijdsverschil tussen een uitgestuurde golf en de detectie van zijn weerkaarsing. Naast de acoustische variant, is er ook een electromagnetische. De laatste zijn nauwkeuriger dan de eerste en de ontwikkelingen van de laatste jaren

hebben producten op de markt gebracht die toelaten ook op korte afstand (grootte-orde $1m$) metingen te doen [1].

- Triangulatie met camera's: twee camera's die vanuit licht verschillend oogpunt kijken laten toe om de diepte in te schatten omdat je de zijden van de driehoek kan uitrekenen tussen het punt in de scene en de overeenkomstige punten in beide beelden. Hetzelfde stereoprincipe kan ook toegepast worden met één camera die beweegt: de camera's zijn dan verschillend in de tijd in plaats van in de ruimte. In elk geval zijn er visueel herkenbare elementen nodig in de beelden, die heten lokale descriptoren. Die worden automatisch uit het beeld gehaald door detectie-algoritmes als Susan, STK, Sift ...

- Triangulatie met gestructureerd licht: werkt volgens hetzelfde principe, alleen in een van de camera's vervangen door een projector, een inverse camera. Die zorgt voor de nodige visuele herkenningspunten die nu zo geconditioneerd kunnen worden dat ze in het camerabeeld gemakkelijk herkend kunnen worden. Een laser die één punt schijnt is een voorbeeld van een projector, dit is 0D gestructureerd licht. Dat punt kan bewegen zodat na een hele videosequentie de hele scene gereconstrueerd is. Liever zouden we het beeld met een videobeeld kunnen reconstrueren: daarvoor kan een beeld vol evenwijdige lijnen geprojecteerd worden met een dataprojector. Dit 1D gestructureerd licht steunt op epipolaire geometrie: de geprojecteerde lijnen staan bij benadering loodrecht op een vlak tussen camera, projector en scene. Om hiervan gebruik te kunnen maken moeten de camera en projector een vaste oriëntatie hebben ten opzichte van elkaar. Als dat niet zo is, is 2D gestructureerd licht een handiger aanpak: individueel herkenbare elementen – bijvoorbeeld cirkels – worden geprojecteerd om een willekeurige, potentieel veranderende oriëntatie aan te kunnen.

- Er zijn nog heel wat andere reconstructietechnieken, die bijvoorbeeld de vorm afleiden uit het silhouet dat het object in het camerabeeld maakt. Of diepteinformatie extraheren door verschillende focuslengtes in te stellen op de camera. Andere technieken gebruik schaduwen of textuur om de vorm af te leiden. Interferometrie is een laatste mogelijkheid.

2 Encoderen

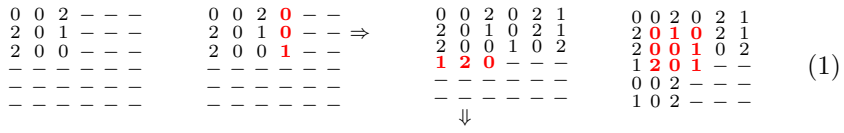
2.1 Patroonlogica

Deze sectie begint met uit te leggen waarom de opstelling die we bestuderen een is waarin de projector een vaste plaats heeft en de camera meebeweegt. De technologie van een doordeweekse dataprojector (met een gloeilamp of gasontladingslamp) laat niet toe de projector te bewegen. Het zou evenwel de

[1] www.mesa-imaging.ch

calibratie veel eenvoudiger maken als dat wel zou kunnen. Vandaar dat de recent opgekomen alternatieven van belang zijn: LED- en laser-projectoren laten bewegingen wel toe. Bij LED is de lichtopbrengt relatief zwak, dus moet de omgeving voldoende donker gemaakt kunnen worden. Als dat niet kan, biedt laser soelaas: daar kunnen de lichtbron en het element dat het licht juist verdeelt over het projectieoppervlak, ontkoppeld worden. Die ontkoppeling laat toe om de laser zelf statisch in de buurt van de robot op te stellen, en de spiegels die het licht verdelen aan de eindeffector te bevestigen.

Een matrix van individueel herkenbare projectie-elementen is een nuttige techniek in deze context. De visieverwerking en de robotcontrole zijn toepassingen die veel rekenkracht vereisen. Daarom is het zinvol om de online verwerking zoveel mogelijk te ontlasten door alle taken die op voorhand kunnen gebeuren op voorhand te doen. Daarom kiezen we voor patronen die herkend kunnen worden onder gelijk welke hoek: dan hoeft er tijdens de beweging van de robot niet voortdurend bijgestuurd worden onder welke hoek de patronen moeten bekeken worden om zinvol gedecodeerd te worden. Vandaar stellen we een nieuw algoritme voor om de logica in dergelijke patronen te berekenen. Het baseert zich op een bestaand algoritme door Morano et al. [1998], dat brute (reken)kracht gebruikt om tot een oplossing te komen. Morano et al. gaat als volgt te werk, genomen dat elke submatrix een grootte 3×3 heeft, zie "vergelijking" 1 : Eerst wordt de submatrix links bovenaan willekeurig gevuld. Dan worden er telkens 3 elementen in de eerste drie rijen toegevoegd zodat elke submatrix maar één keer voorkomt. Dat gebeurt met willekeurige getallen tot er een unieke combinatie gevonden wordt. In een derde stap gebeurt dat met de eerste drie kolommen. In een vierde stap wordt er telkens maar één element toegevoegd om de rest van de matrix te vullen.



(1)

We wijzigen deze methode als volgt:

- Het toevoegen van rotatie-invariantie: een matrix van projectie-elementen impliceert dat er vier buurelementen zijn die dichterbij staan, en vier – de diagonalen – die een factor $\sqrt{2}$ verder staan. Die twee groepen kunnen dus uit elkaar gehaald worden in het camerabeeld. Vandaar dat het volstaat om bij de controle of het patroon wel goed is, elk van de submatrices te draaien over 90, 180 en 270° en te vergelijken met alle anderen.

- Het patroon hoeft niet vierkant te zijn. Veel projectorschermen hebben een 4 : 3 aspectverhouding, en de kiezen de verhoudingen van het patroon dus ook zo. Definieer de verhouding van de breedte op de hoogte als s.

- De grootte van de matrix is niet op voorhand bepaald. We lossen dit probleem recursief op: de oplossing voor een matrix met grootte $n \times \lfloor sn \rfloor$,

wordt gebruikt om een oplossing te vinden voor een matrix met grootte $(n+1) \times \lfloor s(n+1) \rfloor$

- Het probleem krijgt meer structuur door ervoor te zorgen dat het onmogelijk is een van de kandidaat-patronen meermaals te onderzoeken. Alle getallen van de matrix achter elkaar zetten geeft een lang getal dat de toestand van de matrix volledig omvat. Tijdens het zoeken naar een matrix die voldoet aan alle voorwaarden, is dat getal strikt stijgend. De manier van zoeken in deze zoekruimte is eerst in de diepte, waarbij telkens de elementen verhoogd worden die in die stap verhoogd kunnen worden, todat een unieke combinatie is gevonden die voldoet aan alle beperkingen. Als de huidige tak geen oplossing meer kan bieden, dan wordt er gezocht met achterwaarts redeneren (*backtracking*).

Appendix A bevat de pseudo-code die deze techniek volledig reproduceerbaar maakt.
Het is ook mogelijk om een patronen te genereren volgens een honingraatpatroon. Hetzelfde algoritme wordt daarvoor gebruikt, alleen aangepast aan de nieuwe organisatie. De bekomen resultaten zijn in dit geval minder mooi dan in de matrixorganisatie, omdat elk van de elementen nu maar 6 buren heeft in plaats van 8. Dit laat minder vrijheid in de zoekruimte, vandaar de bepertere resultaten. Deze structuur heeft ook voordelen: het is bijvoorbeeld een compactere organisatievorm.

2.2 Patroonimplementatie

Deze code kan nu op verschillende manieren in praktijk worden gezet:

- Vormcodering: Elke letter van het alfabet wordt met een andere vorm geassocieerd. Die vormen worden zo gekozen zodat ze makkelijk uit elkaar te houden zijn bij detectie. Bijvoorbeeld doordat de verhouding van hun oppervlakte tot hun omtrek in het kwadraat, grondig verschillend is. Of doordat hun Fourier descriptor ver uit elkaar ligt.

- Kleurcodering: Als er een discontinuiteit door de vorm loopt, wordt die onherkenbaar, of erger: herkend als een van de andere vormen. Vandaar dat als er geen vormen gebruikt worden, we kiezen voor de meest compacte vorm: de cirkel. Daarmee kan een zo groot mogelijke oppervlakte gegenereerd worden (maximale herkenbaarheid), zo dicht mogelijk bij een centraal punt (minimale discontinuiteitsproblemen).
  Deze codering gebruikt een verschillende kleur voor elke letter, wat zonder aanpassing van het patroon aan de scene, alleen kan bij bijna witte scenes. De keuze van de kleuren is zo dat ze in een kleurenruimte waar helderheid en kleur gescheiden worden, qua kleur zo ver mogelijk uit elkaar liggen.

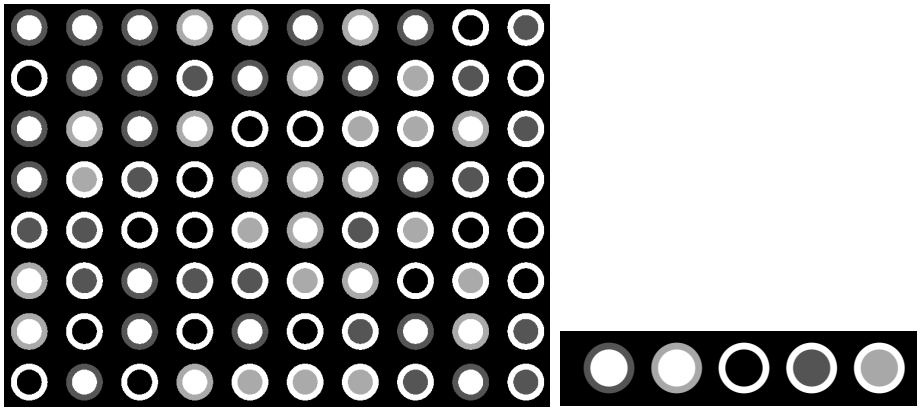
- Illuminantiecodering: In plaats van kleuren, kunnen grijswaarden gebruikt worden. Dat kan wel bij gekleurde scenes. Optische overspraak

is een probleem waarmee blijvend rekening moet gehouden worden: het overvloeien van intensiteit in naburige pixels waarvoor die intensiteit niet bedoeld is.

- Tijdsfrequenties: Patronen kunnen ook variëren in intensiteit in de tijd. Verschillende frequenties en/of fasen zijn mogelijkheden om de letters te encoderen. Dit veronderstelt wel dat elk van de blobs over enkele frames kunnen gevolgd worden, en dat de scene dus niet voortdurend bijzonder snel beweegt zodat tracking onmogelijk zou worden.

- Ruimtelijke frequenties: Een andere mogelijkheid is een sinus van intensiteitsverschillen te verwerken in een cirkelvormige blob. Die intensiteit kan dan best tangentieel verschillen om de hoeveelheid pixels bij elke fase gelijk te houden. Een frequentieanalyse op het camerabeeld levert dan robuust terug de overeenkomstige letters van het alfabet op.

- Relatieve intensiteitsverschillen: Een laatste manier, waar in de rest van de thesis meer nadruk op ligt dan op de anderen, is werken met locale intensiteitsverschillen. Elke blob bestaan dan uit twee grijswaarden, waarvan één dichtbij de maximale intensiteit van de camera zit, zie figuur 2. Aangezien de weerkaatsingseigenschappen in de scene lokaal sterk kunnen variëren, is het nuttig om de detectie van de grijswaarden in het camerabeeld niet absoluut te doen. Een relatieve vergelijking van de ene grijswaarde ten opzichte van de andere binnen eenzelfde element, schakelt de storende factor van verschillende weerkaatsingseigenschappen van het materiaal uit. We kiezen voor concentrische cirkels, dat heeft als extra voordeel dat het systeem ook blijft werken als het camerabeeld wat uit focus is (het middelpunt blijft bijna hetzelfde).



Figuur 2: Links: patroonimplementatie met concentrische cirkels voor $a = h = 5$, $w = 3$; rechts: de voorstelling van de letters $0, 1, 2, 3, 4$

2.3 Patroonaanpassing

Om de robottaak beter ten dienste te zijn, is het nuttig het patroon tijdens de beweging aan te passen:

- Aanpassing in positie: De plaatsen waarvan de diepte moet ingeschat worden zijn afhankelijk van de noden van de robottaak op dat moment. Het beeld van een dataprojector kan op elk moment aangepast worden, en daar kunnen we gebruik van maken. We verschuiven de projectie-elementen naar de plaatsen waar ze meest nodig zijn, ermee rekening houdend dat de buren van een element dezelfde buren moeten blijven.

- Aanpassing in grootte: terwijl de robot – en dus de camera – beweegt, worden de geprojecteerde elementen groter of kleiner. Ze behoren niet groter te worden dan nodig is voor de segmentatie in het camerabeeld. Gebruik maken van een zoomcamera is hier ook interessant. Natuurlijk moeten dan ook de intrinsieke waarden van de camera aangepast worden tijdens de beweging naargelang de aanpassing in zoom.

- Aanpassing in intensiteit: Gezien de potentieel verschillende weerkaatsingsfactoren van de materialen in de scene, zullen voor eenzelfde intensiteit in het projectorbeeld, verschillende intensiteiten in het camerabeeld gevonden worden. Onder- of overbelichting in de camera moeten vermeden worden om de decodering mogelijk te houden. Ideaal zou zijn dat het meest heldere deel van elk van de elementen in het camerabeeld bijna een maximum stimulus oplevert. Dan is de resolutie in intensiteitswaarden in het camerabeeld het grootste, en de relatieve verhouding van intensiteiten het nauwkeurigst. In de mate dat daar rekentijd voor over is, passen we dan ook de intensiteiten in het projectorbeeld individueel aan.

- Aanpassing aan de scene: Als er concrete modelinformatie is over de scene, is het nuttig om een patroon te kiezen dat inspeelt op die informatie.

3 Calibraties

3.1 Intensiteitscalibratie

Zowel camera als projector reageren niet lineair op het invallende licht. We gebruiken een bestaande techniek [Debevec and Malik, 1997] om die curves te identificeren: die maakt voor de camera gebruik van foto's van dezelfde scene met verschillende sluitertijden. Door de projector, als inverse camera, gaat het om verschillende lichtintensiteiten. Deze techniek vermeldt niet welke punten in de beelden moeten gekozen worden om de responsies te identificeren. We stellen daarom een algoritme voor om punten te kiezen die zorgen voor een zo breed mogelijke exitatie van het systeem.

3.2 Geometrische calibratie

3.2.1 Intrinsieke parameters

Zowel camera als projector gebruiken een aangepast *pin hole*-model. Het model wordt in beide gevallen aangepast voor radiale distorties. Voor de dataprojector is er een extra modelaanpassing, aangezien die ontworpen zijn om naar boven te schijnen, wat handig is voor presentaties, maar minder voor deze toepassing.

3.2.2 Extrinsieke parameters

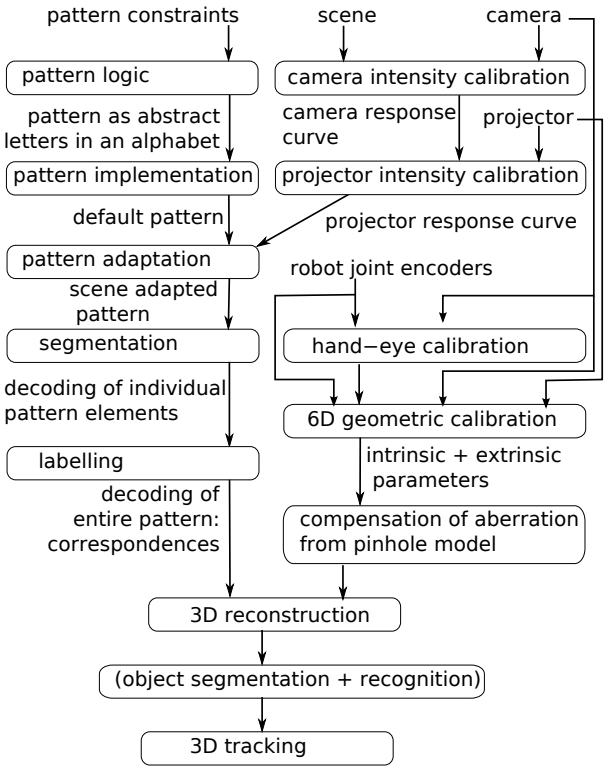
Gezien de beperkte robuustheid van calibraties met een calibratieobject, en de omslachtigheid ervan, kiezen we voor een zelfcalibratie. Bij de technieken voor zelfcalibratie zijn er die enkel werken bij vlakke scenes, en anderen die enkel werden bij niet-vlakke scenes. Deze thesis gebruikt een tussenvorm die met beiden kan werken. De calibratie maakt waar mogelijk gebruik van de extra positiekennis die er is over de camera, aangezien die aan de eindeffector van de robot bevestigd is. Het algoritme wordt in detail beschreven.
Tijdens de beweging van de robot verandert de relatieve positie van camera en projector, en moeten de calibratieparameters aangepast worden. Dat kan door predictie-correctie. In de predictie worden de parameters aangepast aan de hand van de encoderwaarden van de robot. In een correctiestap worden metingen uit het camerabeeld gebruikt (de correspondenties) om met behulp van een techniek als *bundle adjustment* de diepte reconstructie beter te maken.

4 Decoderen

4.1 Segmentatie

Het beeld wordt gesegmenteerd door te steunen op de vaststelling dat de door de projector niet-belichte delen sterk onderbelicht zijn ten opzichte van de belichte delen. Zo kunnen achtergrond en voorgrond van elkaar gescheiden worden. Vervolgens wordt er een histogram gemaakt van de intensiteitswaarden in elk van de geprojecteerde elementen: die kan benaderd worden door een multimodale verdeling met twee Gaussianen, aangezien elk projectie-element twee intensiteiten bevat. De verhouding van de gemiddelde waarden van elk van die twee Gaussianen is bepalend voor het decoderen van welk soort element in het projectorbeeld deze pixels in het camerabeeld afkomstig zijn. De regel van Bayes wordt hier over de hele lijn toegepast om tot een *Maximum A Posteriori*-beslissing te komen.

Het voorgaande gaat over de initialisatie van de verschillende beeldelementen. Terwijl de robot beweegt, willen we deze relatief zware procedure niet telkens doorlopen. Het is efficiënter om een volgalgoritme te gebruiken om de elementen zoveel mogelijk doorheen de tijd te volgen. De tekst vergelijkt verschillende bruikbare algoritmen en kiest voor *CAMShift* voor dit relatief eenvoudige volgprobleem.



FIGUUR 3: Overzicht van de verschillende stappen voor 3D-reconstructie

4.2 Etikettering

Na de segmentatie staat er vast van welk type elk van de elementen in het camerabeeld zijn. Etikettering gaat op zoek naar de juiste buren voor elk van die elementen: hoe zijn die elementen verweven in een grafe. Niet alleen worden de 8 buren van elk element uit het beeld gehaald, ook wordt er gecontroleerd of de grafe wel consistent is: zijn de verschillende buurrelaties wel wederkerig?

4.3 3D-reconstructie

Het eigenlijke reconstrucie-algoritme is een bestaande, vrij eenvoudige, techniek. Het komt neer op het oplossen van een overgedetermineerd stelsel van lineaire vergelijkingen. Deze sectie bevat ook een uitgebreide foutenanalyse: welke van de calibratieparameters zijn in welk bereik gevoelig voor fouten? Waar worden met andere woorden de bestaande fouten versterkt, zodat de kwaliteit van de reconstrutie niet meer aanvaardbaar is. Deze configuraties kunnen dan vermeden worden door ze als een extra beperking bij de controle van de robot te steken (bijvoorbeeld met behulp van *beperkingsgebaseerde taakspecificatie*).

5 Robotcontrole

In deze sectie worden de controle van een robotarm op basis van gestructureerd licht duidelijk gemaakt aan de hand van enkele voorbeelden. Die illustreren hoe de techniek van *beperkingsgebaseerde taakspecificatie* ook van toepassing is op deze sensor. Op die manier kunnen de beperkingen afkomstig van verschillende sensoren geïntegreerd worden.

6 Software

Deze sectie beschrijft het software-ontwerp. De verschillende onderdelen zijn zo modulair mogelijk opgebouwd, zodat de bouwblokken bijvoorbeeld ook voor andere systemen bruikbaar zijn. Om onnodig extra werk te vermijden, wordt gesteund op bestaande bibliotheken. De afhankelijkheden van andere softwaresystemen zitten in *wrappers*: het volstaat de interface-klassen aan te passen om het hele systeem aan te passen aan een alternatieve bibliotheek.

Verder worden de tijdsvereisten voor het systeem bekeken. Er zijn tijdsvertragingen die inherent zijn aan de verschillende onderdelen van de gebruikte hardware. Daarnaast is er de tijd die nodig is om de visie- en controleverwerking te berekenen. Er wordt uitgerekend welke een veilige ware-tijds-frequentie is: die is afhankelijk van de processorkracht. Als die niet genoeg is, worden verschillende hard- en softwareoplossingen voorgesteld om binnen de nodige tijdslimieten te blijven.

7 Experimenten

Dit hoofstuk bevat experimenten met drie robotopstellingen waar gestructureerd licht nuttig is:

- Het **ontbramen van omwentelingslichamen**: Industriële metalen stukken – die ook omwentelingslichamen zijn – hebben last van speculaire reflectie bij het gebruik van gestructureerd licht. Er wordt hier dan ook een gestructureerd-licht-techniek toegepast die compenseert voor speculaire reflecties. Het algoritme vindt alle nodige geometrische parameters van het voorwerp om de braam automatisch te verwijderen.
- Het **automatiseren van een endoscopisch instrument**: In dit experiment wordt een instrument om laparoscopisch te hechten, geautomatiseerd. Na de pneumatische automatisering van een anders manueel stuk gereerdschap, wordt beschreven hoe gestructureerd licht hier nuttig is. Menselijke organen hebben namelijk weinig natuurlijke visueel herkenbare elementen. Er wordt gewerkt met een kunststof vervangvoorwerp voor deze haalbaarheidsstudie. Het gestructureerde licht haalt er de wondrand uit: daarvoor wordt een combinatie gemaakt van 2D- en 3D-visietechnieken.
- **Objectmanipulatie**: Een ander experiment test de 2D projectie-elementen door ruimtelijke nabijheid. Het testobject in de scene is een willekeurig gekromd oppervlak. De verschillende verwerkingsstappen leiden tot een spaarse 3D reconstructie die door de robot gebruikt wordt om het voorwerp op de gewenste manier te benaderen.