Template Reduction of Feature Point Models for Rigid Objects and Application to Tracking in Microscope Images

Manuel Boissenin

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

February 2009

Sheffield Hallam University

Microsystems and Machine Vision Laboratory

The undersigned hereby certify that they have read and recommend to the Faculty of Arts, Computing, Engineering and Sciences for acceptance a thesis entitled "**Template Reduction of Feature Point Models for Rigid Objects and Application to Tracking in Microscope Images**" by **Manuel Boissenin** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: February 2009

Research Supervisor: ____________________
Dr Balasundram Amavasai

Examining Committee: ____________________
Professor Huosheng Hu

____
Professor Melvyn Smith

____
Professor Christopher Care

Sheffield Hallam University

Date: **February 2009**

Author: **Manuel Boissenin**

Title: **Template Reduction of Feature Point Models for Rigid Objects and Application to Tracking in Microscope Images**

Department: **Microsystems and Machine Vision Laboratory**

Degree: **Ph.D.** Convocation: **February** Year: **2008**

Signature of Author

À mes parents,

à mes grands-parents.

Contents

List of Figures x

List of publications xx

Abstract xxii

1 Introduction 2

1.1 Rationale and motivation . . . 2

1.1.1 The MINIMAN project . . . 3

1.1.2 The MiCRoN project . . . 3

1.2 Research aims and objectives . . . 4

1.2.1 Choices and discussion . . . 5

1.3 Research methodologies . . . 10

1.4 Contributions . . . 10

1.5 Organisation of the thesis . . . 11

2 Literature review 13

2.1 Introduction . . . 13

2.2 Connected techniques to visual tracking . . . 15

2.2.1 Detection and initialisation . . . 15

2.2.2 Background subtraction and movement detection . . . 16

2.2.3 Mathematical Morphology . . . 16

2.2.4 Techniques to match point sets . . . 17

2.2.5 Image distortion . . . 21

2.3 Visual tracking . . . 23

2.3.1 Kalman filter based visual tracking . . . 25

2.3.2 Particle filter based visual tracking . . . 27

2.3.3 Other visual tracking algorithms 29

2.4 The Hough transform 32

2.4.1 Introduction 32

2.4.2 Hough transform and tracking 33

2.4.3 The randomised Hough transform 34

2.5 Summary 37

3 Particle filters **38**

3.1 Introduction 38

3.2 A real world example 39

3.3 Typical encountered issues 40

3.4 Preliminary approaches to solve these issues 41

3.4.1 Dynamic template updating 43

3.4.2 Taking measures 44

3.4.3 Improving the matching paradigm 46

3.4.4 Geometric Branch-and-Bound Matching 48

3.5 Clustering particles 50

3.5.1 Locating the statistical modes 50

3.5.2 Method formalisation 51

3.5.3 An $O(n)$ algorithm to cluster particles 51

3.5.4 Location of the micro-pipette using further measurements of the image 54

3.6 Further improvements 56

3.6.1 Overcoming the effect of clutter 56

3.6.2 Integrating the kinematics of the object 56

3.6.3 Partial re-initialisation 57

3.6.4 Modes filtering 58

3.7 A generic particle filter architecture 59

3.7.1 Description of the architecture 59

3.7.2 Issues related to the initialisation of the class 59

3.7.3 Conception 60

3.7.4 Extensions 62

3.8 Summary 63

3.8.1 Possible improvements 64

4 Shape information and template reduction 65

4.1 Introduction 65

4.1.1 Square example 65

4.1.2 Circle example 69

4.1.3 Locating a shape on an image 71

4.2 Self similar set of points 75

4.3 Robustness to noise 78

4.4 Template reduction, a simple example 79

4.5 Evaluation of the characterising value of a set of points 79

4.6 A generic algorithm for the pose estimation of rigid objects 82

4.7 Summary 84

4.7.1 Future research 85

5 The stencil estimator 86

5.1 Introduction 86

5.2 The stencil estimator 86

5.3 Robustness 91

5.4 Implementation of the tracking algorithm 92

5.5 Stencil reduction 96

5.6 Summary 98

6 Experiments 100

6.1 Tracking of Microscopic Objects 100

6.1.1 Context and experimental setting 100

6.1.2 Algorithm 103

6.1.3 Complexity analysis 105

6.1.4 Parallelisation 110

6.1.5 Summary 111

6.2 Testing of the stencilled Hough transform using synthetic data 112

6.2.1 Experiments and results 112

6.2.2 Comparison with different similarity methods using an exhaustive search 132

6.2.3 Summary 142

6.3 Testing the particle filter on the micro-pipette tracking sequence . . . 143

6.4 Testing the stencil Hough transform on the pipette video sequence . . 151

6.4.1 Experiments and results . . . 151
6.4.2 The motion model . . . 153
6.4.3 Summary . . . 156

7 Conclusion **159**
7.1 Contributions . . . 159
7.2 Future research . . . 160

A Fitting a square to a set of points **162**
A.1 3 points case . . . 162
A.2 4 points case . . . 163
A.3 5 points . . . 167
A.4 Perspective transformation . . . 169

B Dense disparity map using epipolar geometry **171**
B.1 Depth maps of 3-D scenes . . . 171
B.1.1 Stereo vision . . . 171
B.1.2 Camera calibration . . . 172
B.1.3 Stereo calibration . . . 175
B.1.4 Correspondence problem . . . 175
B.1.5 Discussion . . . 177
B.1.6 Summary . . . 178
B.2 Model building alternatives . . . 179
B.3 Implementation . . . 180
B.4 Triangulation . . . 181
B.5 Summary . . . 183

C Further testing of the particle filter **184**
C.1 Table tennis sequence . . . 184
C.2 Rubik's cube sequence . . . 184

D Machine vision and computer vision **191**

E Tracking source code **193**

F Extended abstract **200**

List of Figures

1.1 An artist impression of the MiCRoN project made for the project proposal: mini robots, 1 cm across, working cooperatively in an assembly task 4

1.2 Overview of the MiCRoN set up during the integration stage. The camera is mounted on a Miniman robot. In the front, an infrared communication device is mounted on a USB port. 4

1.3 A tethered MiCRoN robot with its gripper under the microscope camera. 5

1.4 The set up at Sheffield Hallam university, the camera support is mounted on a piezo-electric three degree of freedom translation stage. Green LEDs are used to illuminate the scene. Green has been chosen to avoid interferences with the infrared communication of the MiCRoN robots. 1 pixel in the obtained image corresponds to 1 micrometre (μm). 6

1.5 The graphical user interface of the recognition and tracking software. The *x*-*y* axis frame indicates the position of the recognised and tracked syringe chip. The image has been taken with a optical microscope and its quality is better than the images obtained with the MiCRoN camera. 7

1.6 The MiCRoN robot. 7

2.1 By obtaining the intrinsic parameter it is possible to determine where each point expressed in the camera frame will be projected onto the image. 22

2.2 A red parallelepiped and a white line projected on the image using the image formation model 24

2.3 The right image is the rectified left image. Notice that the power supply edge highlighted in previous figure appears straight. 24
2.4 The Kalman filter can be divided into 2 stages 25
2.5 Diagram to explain the Hough transform, see text for explanation. . . 33

3.1 Particle filtering stages. 39
3.2 The original pen tip tracking algorithm using particle filters [1]. . . . 40
3.3 The tracking of the pipette tip has failed. The white square is the tracked location (at the top centre position of the image). Top left hand corner is the template image. 41
3.4 Problematic frames from the micro-pipette sequence. 42
3.5 Tracking with the template updating mechanism running 43
3.6 Image showing the value of the correlation measure within a region of interest. The positions with high correlation values are coloured following the colour code described in page 44. 45
3.7 Frames showing the tracking results using edge template matching. The top left corner of images shows the edge template of the micro-pipette tip. 47
3.8 The same image filtered with the Susan and the Canny edge detector. 48
3.9 Frames showing the results of edge correlation with only part of the scene image processed in order to filter edges. Frame B illustrates one of the frames where edge correlation fails. 49
3.10 In frame A points represents particles. Frame B shows the thresholded particles. In frame C, the boundaries show how these particles are grouped and can be seen as the set bounds . Frame D shows how the other particles are collated into the corresponding sets. Frame E shows the resulting peak points, compared with frame B the modes are shifted from the thresholded particles and the 2 closed thresholded particles have been integrated in the same mode. 52
3.11 Tracking of the micro-pipette tip. Modes give an indication of the location of the peaks of the *pdf* sampled by the particles. 55
3.12 Tracking stages. 58
3.13 UML diagram of the first design of the particle filter 61
3.14 UML diagram of the final implementation of the particle filter 62

4.1 Squares sharing a common point. 66
4.2 The three squares, modulo rotations, having these two points as corners. 67
4.3 Two possible configurations of 3 points and example of squares that matches these configurations. On the left, 2 points are closer whereas on the right, points are equidistant. 67
4.4 The points a,b,c and d uniquely define a square. 68
4.5 An impossible configuration and 2 possible configurations of 4 points that fit more than one square. 69
4.6 A configuration of 5 points that can be fitted by only one square. . . . 70
4.7 A template (S), its associated state space (T) and the corresponding look-up table (L). Colours are used to represent the states referenced by the look-up table. 80
4.8 Figures A-D show non ambiguous reduced templates. Figures E,F show ambiguous reduced templates 80

5.1 The stencils of the "star" shape corresponding to the coloured area of the transformation space have been drawn in the image space. The coloured regions are the set of points overlapped by the shape when it is moved according to the transformations of the subset having the same colour in the transformation space. 87
5.2 Pre-processing stage 93
5.3 Tracking stage 95

6.1 Images from the gripper tracking sequence. The grippers in the left and right images are at different depths. 101
6.2 Diagram of the set-up used for the experiments. A photo of the set-up can be found figure 1.4, page 6. 101
6.3 A subset of the stack of images that serves as the gripper model . . . 102
6.4 A sample of some of the stencils produced 105
6.5 Key for figure 6.6 107
6.6 Tracking using 4 sets of parameters, see figure 6.5 for keys and figure 6.1.3 for the corresponding tracking behaviour 108
6.7 Speed comparison of the tracking of the gripper using different parameters 108
6.8 Visual comparison of the asymptotic tracking behaviour 109

6.9 Number of stencil increment versus number of image feature 109
6.10 Comparison of Qt threads and boost threads, both implementation uses Qt mutexes 110
6.11 Comparison of the parallelisation on 2 processors using different implementations 111
6.12 Speed comparison, one thread versus two threads 112
6.13 Templates of the tracked shapes. 114
6.14 Appearance of the first image of the hand shape tracking sequence for different noise levels. In percentage of image noise: 0, 4, 10, 16, 22, 28, 34, 40, 46 and 50% respectively 115
6.15 Appearance of the first image of the watch tracking sequence for different noise levels. In percentage of image noise: 0, 4, 10, 14, 20, 26, 30, 34, 38, 42, 46 and 50% respectively 116
6.16 The z-axis corresponds to the number of images where the tracking result differs from the ground truth. The ratio of kept elements of the stencil is actually the ratio of the number of references listed in the 2-D array corresponding to the stencil elements kept. The colour lines outline error levels. This graph corresponds to the hand shape image sequences. 117
6.17 The same graph as in figure 6.16 viewed from the top and with just the error line levels. The levels are the number of images out of the 500 images of the hand shape sequences where the tracking fails . . . 118
6.18 The z-axis corresponds to the number of images where the tracking result differs from the ground truth. The ratio of kept elements of the stencil is actually the ratio of the number of references listed in the 2-D array corresponding to the stencil elements kept. The colour lines outline error levels. This graph corresponds to the watch shape sequences 119
6.19 The same graph as in figure 6.18 viewed from the top and with just the error line levels. The levels are the number of images out of the 800 images of the watch shape sequences where the tracking fails . . . 120

6.20 Histograms characterising the 2-D array containing stencils generated off-line for the hand shape. Each row corresponds to a different tuning of the area size of the stencils. These are 100 %, 56 %, 32 % and 18% of the references respectively 123
6.21 Histograms characterising the 2-D array containing stencils generated off-line for the hand shape. Each row corresponds to a different tuning of the area size of the stencils. 11 %, 6 %, 3% and 1% of the references respectively 124
6.22 Stencils for the hand shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to those of figure 6.20: 100%, 56%, 32%, 18% . . 125
6.23 Stencils for the hand shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to those of figure 6.21 (11%, 6%, 3%, 1%) and each column a different stencil 126
6.24 Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to those of figure 6.20. Each column represents a different image from the hand image sequence . . 127
6.25 Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.21. Each column represents a different image from the hand image sequence 128
6.26 What has been termed average robustness is, in fact, the average difference of, the number of elements of a stencil, and, the number of overlapping elements of its most overlapping stencil 129
6.27 By total robustness we refer to the minimum, for all stencils, of the robustness such it is explained in figure 6.26 129
6.28 Time taken to track a shape versus the level of noise and the decimation ratio of the stencils for the hand image sequences 130
6.29 The graph of figure 6.28 from a different viewpoint. This shows how the level of noise affects the tracking time 130
6.30 The graph of figure 6.28 from a different viewpoint. This shows how the stencil decimation ratio affects the speed performance of the tracking131

6.31 Histograms characterising the 2-D array containing stencils generated off-line for the watch shape. Each row corresponds to a different tuning of the area size of the stencils. These are 100 %, 80 %, 60 % and 30% of the references respectively 133
6.32 Histograms characterising the 2-D array containing stencils generated off-line for the watch shape. Each row corresponds to a different tuning of the area size of the stencils. These are 10 %, 5%, 3% and 1% of the references respectively 134
6.33 Stencils for the watch shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to figure 6.31 135
6.34 Stencils for the watch shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to figure 6.32 136
6.35 Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.31 . Each column represents a different image from the watch image sequence 137
6.36 Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.32 . Each column represents a different image from the watch image sequence 138
6.37 What has been termed average robustness is, in fact, the average of the difference of, the number of elements of a stencil, and, the number of overlapping elements of its most overlapping stencil 139
6.38 By total robustness we refer to the minimum, for all stencils, of the robustness such it is explained in figure 6.37 139
6.39 Time taken to track a shape versus the level of noise and the decimation ratio of the stencils for the watch image sequences 140
6.40 The same graph as in figure 6.39 but with a different viewpoint. This shows how the level of noise affects the tracking time 140
6.41 The same graph as in figure 6.39 but with a different viewpoint. This shows how the stencil decimation ratio affects the speed performance of the tracking 141
6.42. The template is shown on the top left corner. The small black dots are the particles. The centre of the square is the tracked location. . . 145

6.43 The weight measures for this graph and the subsequent graphs represents edge correlation measure. This is an ideal case where the measure gives a unimodal *pdf.* x and y axis are the pixel coordinates of the image point measured. It can be observed that correlation measure is well localised which is a disadvantage for particle filters that are likely to sample the tracked object only on the neighbourhood of the object. 145
6.44 In spite of the blurred features of the pipette, its location is found. Big black square dots are the peak points. 146
6.45 The heavy background clutter is illustrated by the existence of multiple peaks in the graph. In spite of the heavy clutter the pipette tip is well localised through further evaluation of the peak points. 146
6.46 A rare case where the tracking has failed. The centre of the square is the tracked location. 147
6.47 The graph illustrates that most of the peak point obtained, though the correlation of the edge template, are located around the pipette tip. But subsequent multivariate feature measure picked up the wrong peak point as the probable location. 147
6.48 Another cluttered image, this one is due to the change of the background of the pipette and illumination of the scene. The small black square points are the peak points. Centre of the square is the tracked location. 148
6.49 The graph illustrates the background clutter which gives rise to multiple peak points. 148
6.50 Last image of the tracking sequence. Magenta points are previous tracked locations. 149
6.51 Actual location of pipette (line) and tracked location (crosses). 149
6.52 Number of tracked frames versus distance of actual location and tracked location. 150
6.53 Actual location of pipette (line) and tracked location (crosses). Only tracked location of points with high probability is displayed 150
6.54 Two images from the pipette tip sequence. It is the same sequence that was used to discuss the particle filter algorithm. 151

6.55 Comparison of the tracked positions and the manually determined positions of the pipette tip when using our custom edge detector. . . 152
6.56 Comparison of the tracked positions and the manually determined positions of the pipette tip using the Canny edge detector with the first set of parameters. 152
6.57 Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the second set of parameters. 153
6.58 Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the first setting and the motion filter. 154
6.59 Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the second setting and the motion filter. 155
6.60 Tracking accuracy 155
6.61 Tests carried out on an Intel Celeron Northwood 2.7 GHz CPU . . . 155

A.1 An infinite number of squares can fit 3 points. 162
A.2 *abcd* and *abce* are 2 four point convex configurations. 164
A.3 For this configuration of four points we consider the 2 points furthest apart. 164
A.4 The green shaded square was constructed by assuming that the 2 points furthest apart belong to opposite edges and the 2 remaining points to one of the edge. 165
A.5 It is not always possible to fit a square that intersects the 4 points using the assumption that the 2 points furthest apart belong to opposite edges and the 2 remaining points to one of the edge as shown with this configuration of four points. 165
A.6 If the longest distance is an edge of the convex quadrilateral and we assume that the two remaining points belong to different edge of the square there exists at most two squares that fit the four points. . . . 166
A.7 An example when the four points are fitted by a square assuming that the two points furthest apart are on opposite edges of the square and the remaining points belong to different edges. 167

A.8 By considering the direction of intersection, represented by arrows, of the plain black lines the four points can be separated into 3 groups: the point circled in green, the other point on the dashed green line and the two remaining points. 168
A.9 2 Points are taken randomly, by assuming that they belong to opposite edges, for instance, a system of equations can be written and solved to check if it is possible to construct squares that are intersecting this four points. Curved arrows represent the possible rotations of the parallel lines. 168
A.10 8 points, any 3 of them not aligned, in a convex configuration can be fitted by 2 quadrilaterals. 170

B.1 The images seen by each eye are slightly different. *Image from the Optometrists Network website.* 173
B.2 By obtaining the intrinsic parameter it is possible to determine where each point expressed in the camera frame will be projected onto the image. 173
B.3 A red parallelepiped and a white line projected on the image using the image formation model 174
B.4 The right image is the rectified left image. Notice that the power supply edge highlighted in previous figure appears straight. 175
B.5 Top: pair of images taken by our stereo rig. Bottom: the same pair after standardisation. Black lines have been drawn to exhibit the alignment of the corresponding image elements. 176
B.6 Pair of standardised image and their disparity map. Red points are displayed when uncertainty is too high. 177
B.7 Result of our implementation on a well known pair of standardised images from the Tsubaka university repository. 181
B.8 Different disparity maps using our calibrated images. 182
B.9 Triangulation of a point. 182

C.1 Tracking of a ping-pong ball. Top left corner, the ball template image. The large white dots represent the previous tracked positions of the ball. The large black dot, the current position of the ball and the small white dots, the particles' positions. 185

C.2 The large black square is the probable actual position of the ping-pong ball. The small white dots indicate the particles' positions and the larger white squares represent the earlier tracked positions. 185
C.3 After a few frame the tracking fails completely. 186
C.4 Different stage of the image filtering process 187
C.5 The filtered image with false colours. 187
C.6 Tracking a Rubik's cube, transparent white doted lines indicates the hypothesis location. 189

List of publications

1. M. Boissenin, J. Wedekind, A.N. Selvan, B.P. Amavasai, F. Caparrelli and J.R. Travis. Computer vision methods for optical microscopes. Image and Vision Computing, vol. 25, no. 7, p. 1107-1116, Elsevier Science Journal, 2007.

2. M. Boissenin, J. Wedekind, B.P. Amavasai, F. Caparrelli and J. Travis. Fast pose estimation for microscope images using stencils. IEEE Systems, Man and Cybernetics Society 5th Conference on Advances in Cybernetic Systems, p. 49-54, Septembre 7-8, 2006

3. K. C. Lim, M. Boissenin, B.P. Amavasai and R. Saatchi Development of a desktop freehand 3-D surface reconstruction system, IEEE SMC UK&RI 6th Conference on Cybernetic Systems 2008, London, UK.

4. J.Wedekind, B.P. Amavasai, K. Dutton, M. Boissenin A machine vision extension for the Ruby programming language, proceedings 2008 IEEE, International Conference on Information and Automation, 2008, Zhangjiajie, China.

5. J. Wedekind, M. Boissenin, B.P. Amavasai, F. Caparrelli and J. Travis. Object recognition and real-time tracking in microscope images. IMVIP, 2006.

6. B.P. Amavasai, F. Caparrelli, A. Selvan, M. Boissenin, J.R. Travis and S. Meikle. Machine vision methods for autonomous micro-robotic systems. In Kybernetes Journal, vol. 34 no. 9/10 2005, ISSN 0368-492X.

7. M. Boissenin, B.P. Amavasai, J. Wedekind and R. Saatchi. Shape information: using state space to select discriminative configuration of points. BMVA symposium, Shape Representation, Analysis and Perception, November 5, 2007.

8. A. Eisinberg, K. Houston, F. Caparrelli, B.P. Amavasai, M. Boissenin and

P. Dario. Marking techniques for vision recognition of microgrippers for micromanipulation. IEEE International Conference on Robotics and Automation (ICRA2006). Orlando, Florida, May 15-19, 2006.

9. A.N. Selvan, M Boissenin, B.P. Amavasai, F Caparrelli and J.R. Travis. Tracking translucent objects in cluttered scenes. IEEE SMC Chapter Conference on Cybernetics Intelligence–Challenges and Advances, 110-8, 2003.

10. R. Estana, J. Seyfried, M. Thiel, S. Johansson, N. Snis, J.M. Breguet, W. Driesen, T. Velten, J. Gao, P. Vartholomeos, S.G. Loizou, K.J. Kyriakopoulos, M. Boissenin, F. Caparrelli, J. Wedekind, A. Eisenberg, K. Houston, J. Samitier, M. Puig-Vidal, P. Miribel, A. Diéguez and H. Woern. The MiCRoN Project, IST-2001-33567, March 2002 - February 2005.

Abstract

This thesis addresses the problem of tracking rigid objects in video sequences.

A novel approach to reducing the template size of shapes is presented. The reduced shape template can be used to enhance the performance of tracking, detection and recognition algorithms. The main idea consists of pre-calculating all possible positions and orientations that a shape can undergo for a given state space. From these states, it is possible to extract a set of points that uniquely and robustly characterises the shape for the considered state space. An algorithm, based on the Hough transform, has been developed to achieve this for discrete shapes, *i.e.* sets of points, projected in an image when the state space is bounded.

An extended discussion on particle filters, that serves as an introduction to the topic, is presented, as well as some generic improvements. The introduction of these improvements allow the data to be better sampled by incorporating additional measurements and knowledge about the velocity of the tracked object. A partial re-initialisation scheme is also presented that enables faster recovery of the system when the object is temporarily occluded.

A stencil estimator is introduced to identify the position of an object in an image. Some of its properties are discussed and demonstrated. The estimator can be efficiently evaluated using the bounded Hough transform algorithm. The performance of the stencilled Hough transform can be further enhanced with a methodology that decimates the stencils while maintaining the robustness of the tracker. Performance evaluations have demonstrated the relevance of the approach. Although the methods presented in this thesis could be adapted to full 3-D object motion, motions that maintain the same view of the object in front of a camera are more specifically studied.

The work presented in this thesis was funded by the European Union 5^{th} Framework Programme, project No: IST-2001-33567, through the MiCRoN project.

Acknowledgements

I wish to express my sincere gratitude to all the people that have contributed in some way or another to this thesis. I would like to first thank Dr **Jon R. Travis** for giving me the opportunity and all the facilities to carry out this thesis to a successful completion.
I greatly appreciated the excellent supervision of Dr **Bala Amavasai**, his constant encouragements, his most valuable support and his much appreciated leadership by example. They have been constant sources of inspiration throughout my project.
It has been a pleasure to work with Mr **Arul Nirai Selvan**; his commitment to supporting me and his ongoing advice were of a great help.
I am deeply indebted to **Jan Wedekind** for his support and for sharing with me his software expertise. I have learnt a lot from him and the numerous discussions we had helped to shape the development of this thesis.
I would also like to express my gratitude to Dr **Reza Saatchi** for his kind support and his many helpful corrections and suggestions.
Many thanks to the examining committee: professor **Huosheng Hu**, professor **Melvyn Smith** and professor **Christopher Care** for reviewing my work and providing me extremely valuable feedback.
Thanks to **Amir**, **Kim**, **Stephen** and all the ERASMUS students that stayed in the lab as well as all the people I have met in Sheffield who made life taste so much better and with whom I have spent good and great moments.
Finally, thanks to **Susan** who made me the marvellous present of touching my heart.

Chapter 1

Introduction

1.1 Rationale and motivation

"There is plenty of room at the bottom" Richard P. Feynman (1959)

The field of micro-manipulation and micro-robotics is in its very early stage of development and there only exist a handful of useful industrial applications within this area so far. It is predicted that in the near future a wider variety of application fields, from high-precision and fast assembly of mechanical micro-components in industry to the handling of cells in medical or biological applications will require efficient systems for micro-manipulation. Micro-manipulation systems promises many benefits for society as well as very high returns to the players that will create the technology. To operate such systems require generic feedback. Currently, there are six main technologies [2] than can estimate the pose of an object and track its motion:

- mechanical sensing, generally using electro mechanical transducers.
- inertial sensing usually through MEMS (micro-electronic mechanical systems).
- acoustic sensing with ultrasonic waves.
- magnetic sensing using captors that measures the local magnetic field that can be altered by using local sources.
- radio and microwave sensing on the time of flight principle.
- optical sensing using charge-coupled devices (CCDs).

Considering the size of the objects and their environment visual feedback will be a key component in the realisation of micro-manipulation systems. These are amongst the major motivations for the work presented in this thesis.

The origin of the work that is presented here is intimately related to and emerges from the development of two European union funded projects.

1.1.1 The MINIMAN project The main objective of the MINIMAN project [3] was the development of a smart micro-robot with 5 degrees of freedom and a size of a few cubic cm, capable of moving around by the use of tube-shaped multi-layered piezo-actuators. Controlled by visual and force/tactile sensor information, the micro-robot is able to perform manipulations with a motion resolution down to a few nano-metres (nm) in either a tele-manipulated or semi-automated mode. The intention was to free humans from the tedious task of having to handle minuscule objects directly. Equipped with micro-machined grippers, the robot is able to perform high-precision grasping, transportation, manipulation and positioning of mechanical or biological micro-objects, under an optical microscope or within the vacuum chamber of a scanning electron microscope. A powerful computer system using inexpensive PC-compatible hardware components ensured the robot operation was carried out in real-time. The key to closed loop control was vision-based feedback. The vision system was used to locate objects and tools within the workspace. This project was successfully completed and was then followed by the MiCRoN project.

1.1.2 The MiCRoN project The MiCRoN project [4] was a continuation of the MINIMAN project. The MiCRoN project involved eight European partners and its aim was to utilise a co-operative set of micro-robots that could form the basis of a micro or nanofactory (figure 1.1). The work-packages that the Microsystems & Machine Vision Laboratory (MMVL) at Sheffield Hallam University (SHU), UK, was involved in, comprised the design and implementation of the vision control system, position sensors and support in path planning. Figure 1.2 shows the actual set up during the integration phase. In figure 1.3 a close up of the work area is shown. The methods presented in this thesis was used for providing visual feedback using a camera with a magnifying lens and were developed using the set-up shown in figure 1.4. The graphical interface is presented in figure 1.5. Some of the robots, built by the consortium members, are shown in figure 1.6. Powering, actuation and

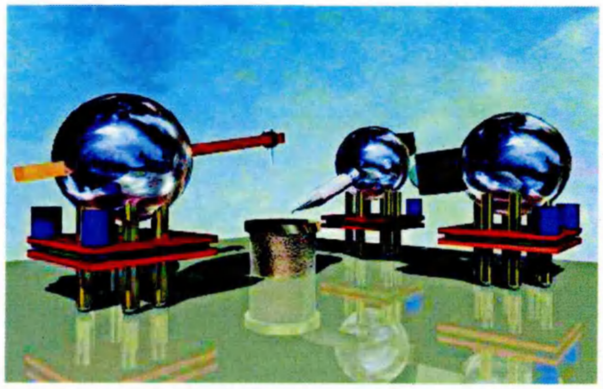


Figure 1.1: An artist impression of the MiCRoN project made for the project proposal: mini robots, 1 cm across, working cooperatively in an assembly task

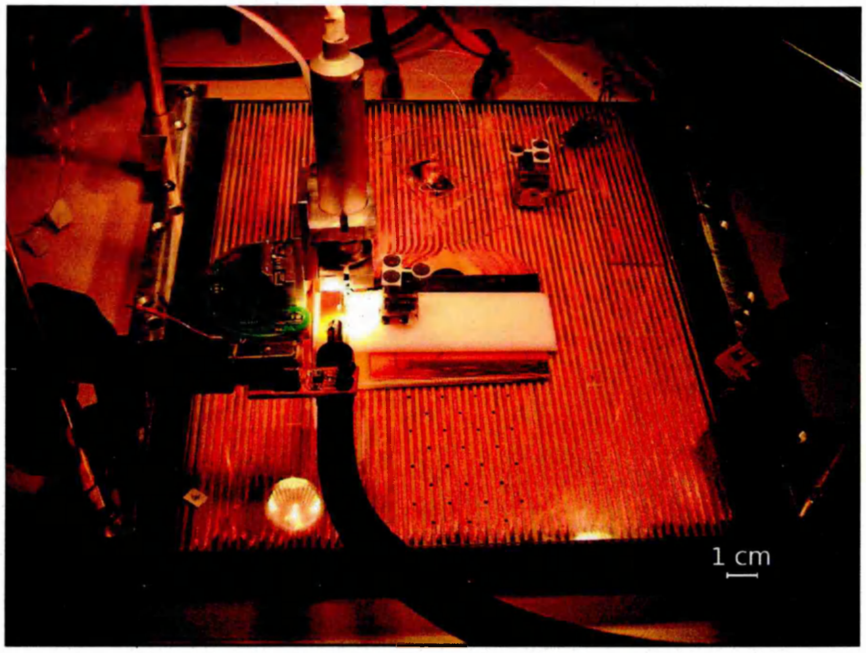


Figure 1.2: Overview of the MiCRoN set up during the integration stage. The camera is mounted on a Miniman robot. In the front, an infrared communication device is mounted on a USB port.

locomotion modules were integrated into a single miniature package.

1.2 Research aims and objectives

One of the main issue of the project was the development and implementation of a vision system to give 3-D visual feedback for the manipulation tasks carried out by the micro robots. Since the manipulation task was not pre-determined, the

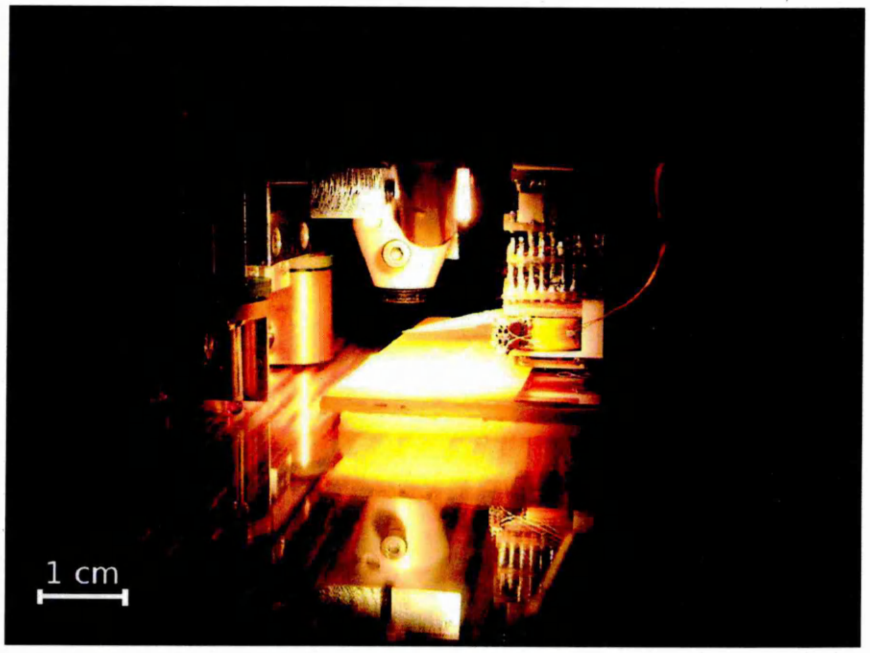


Figure 1.3: A tethered MiCRoN robot with its gripper under the microscope camera.

robot tools needed to be developed and the manipulated objects selected during the course of the project, the delivered vision system had to be generic enough to be adapted to the final objects that were used for the demonstration tasks of the project. Additionally, the pose estimation had to be done in real-time so that it did not become a bottleneck in the overall system. Moreover, since the system that took images was constrained in size, the image quality was impaired and the developed methodology had to be robust to these induced disturbances.

In order to address the real-time issue the focus was first put on tracking. To address the issue of adaptability the model of the object was built using an image of the object, the 3-D pose estimation was done with 4 degrees of freedom –2 sideways translations, orientation and depth of the object– and the robustness issue was dealt with by using a robust similarity measure.

1.2.1 Choices and discussion One of the main topic of this thesis is tracking, an important problem, since it has a wide range of applications (*e.g.* visual feedback, data compression in videos, scene interpretation, *etc*), that has been extensively researched by the computer vision community.

In this thesis, tracking refers to a technique that significantly reduces the search space on where to look for an object using information that can help predict its

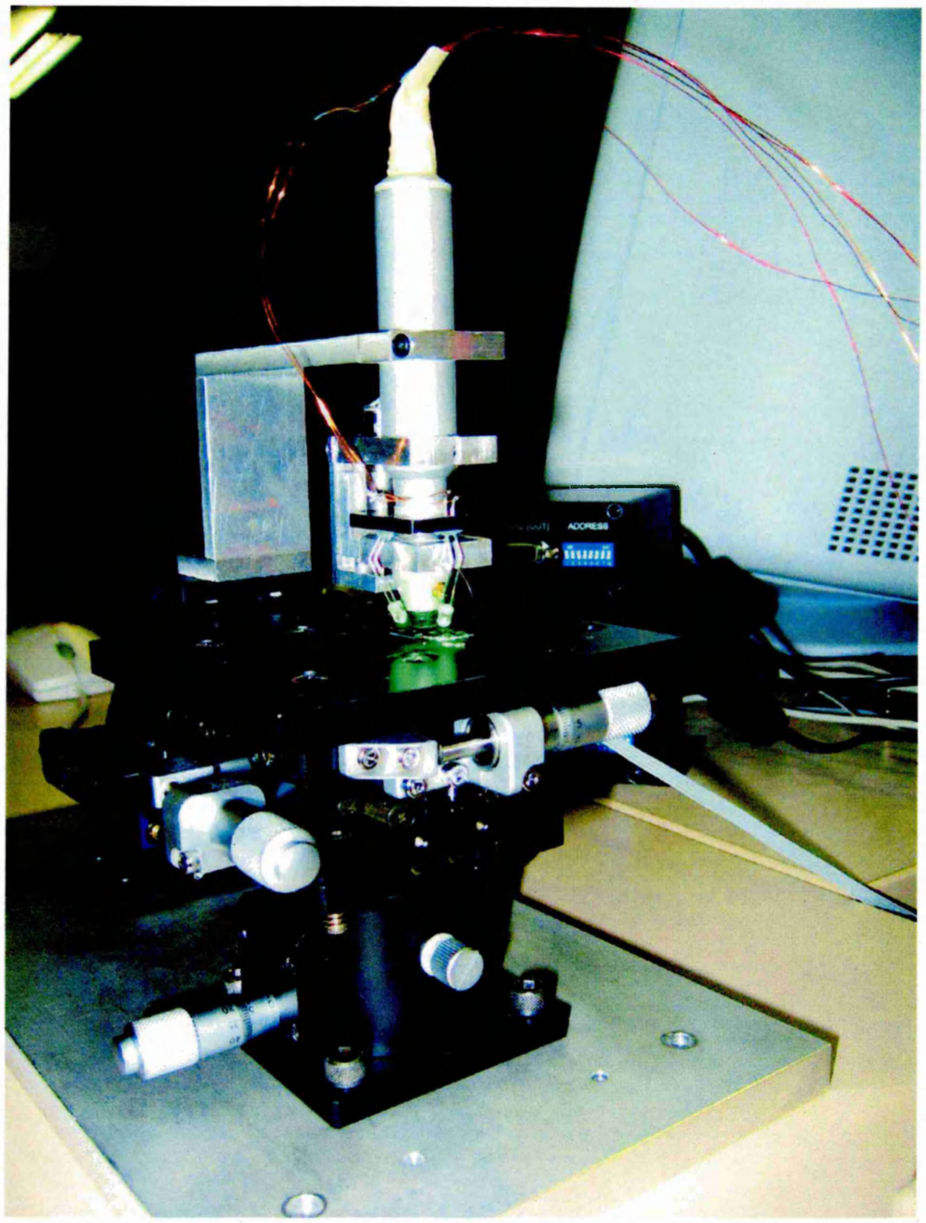
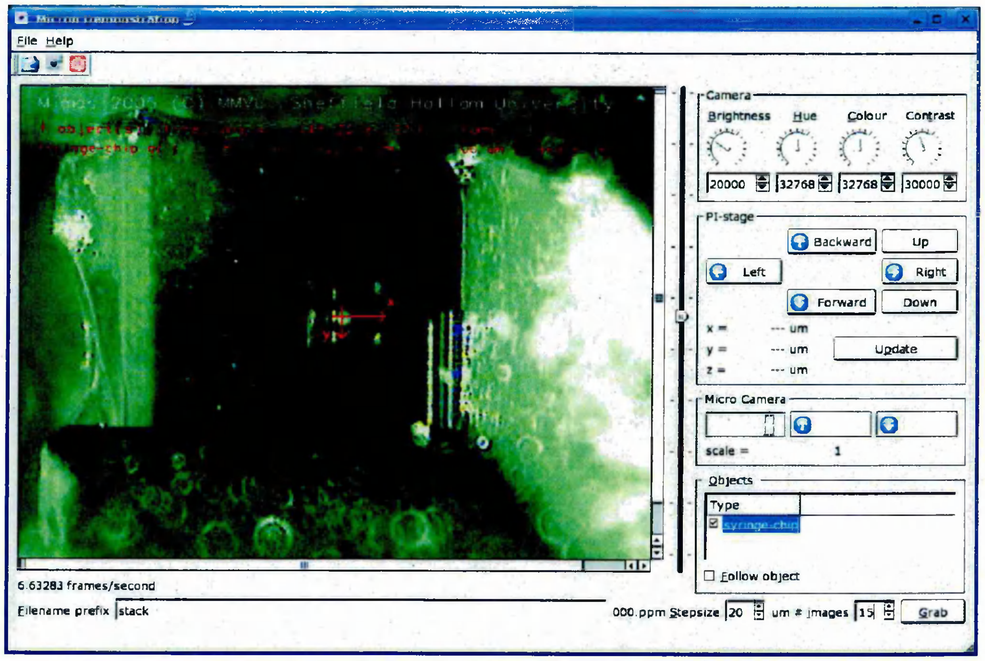


Figure 1.4: The set up at Sheffield Hallam university, the camera support is mounted on a piezo-electric three degree of freedom translation stage. Green LEDs are used to illuminate the scene. Green has been chosen to avoid interferences with the infrared communication of the MiCRoN robots. 1 pixel in the obtained image corresponds to 1 micrometre (μm).

position such as its previous estimated position and speed or the dynamics of the system. Some researchers advocate the terminology time coherence. Taking these considerations into account, object tracking is a process that consists in identifying sets of pixels in an image sequence that correspond to the same object. In contrast, *detection* achieves the same feat but without any prior assumption on the location



Brightness 20000
Hue 32768
Colour 32768
Contrast 30000
x = --- um
y = --- um
z = --- um
scale = 1
Type
☑ syringe-chip
☐ Follow object
6.63283 frames/second
Filename prefix stack
000.ppm Stepsize 20 um # images 15

Figure 1.5: The graphical user interface of the recognition and tracking software. The x-y axis frame indicates the position of the recognised and tracked syringe chip. The image has been taken with a optical microscope and its quality is better than the images obtained with the MiCRoN camera.

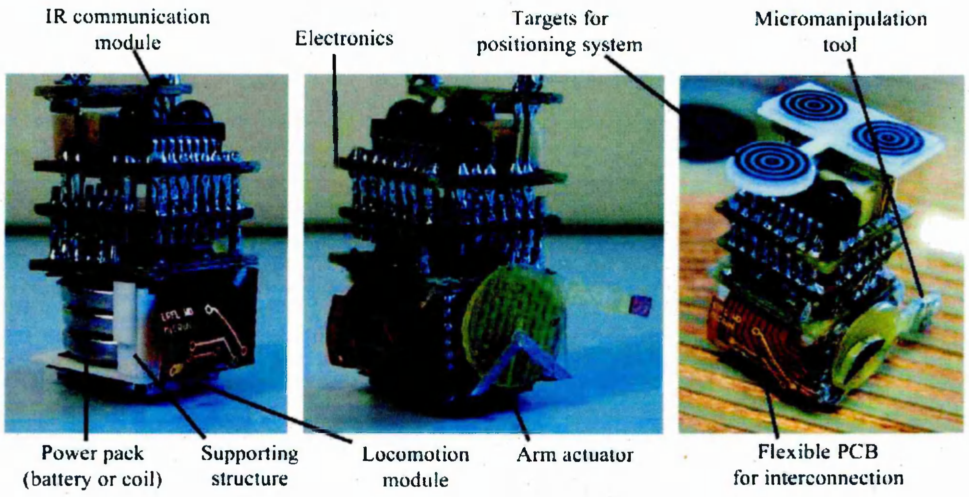


Figure 1.6: The MiCRoN robot.

of the object. Recent improvements on the characterisation of features have made detection a viable alternative for locating objects in real-time. This should not be confused with *recognition* that intends to assign labels to image regions. These labels are generally drawn from a large number of labels representing different object classes.

To estimate the position of an object some knowledge about the object and a way to use this knowledge with the image sequence is required. This leads to two interlinked issues: how to represent the object and how to match this representation in the image. As explained above, it was chosen to model this knowledge by using a sample image of the object, often called a template image, taken in the first frame of the sequence. One way to locate the object in the scene image is to compare all parts of the image with the model *i.e.* to measure the similarity between parts of the image and the template image. At this stage the following issues need to be addressed:

- Clutter: due to the limitation of the chosen measurement method, parts of the image can match well the object model even when the object is not present.
- Occlusion: the tracked object might be partly occluded, extend beyond the image, or, move beyond the image for a few frames. When it becomes visible again the algorithm should be able to locate it again.
- Noise: given a scene that does not change, the image acquisition system produces images that are slightly different. This is often due to limitations in the sensors or electronics. Comparison algorithms should be able to cope with these small variations that can often be modelled by Gaussian noise.
- 2-D images of the 3-D world: the image acquisition process is not a bijective mapping; information about the scene is lost in the process, however supplemental information can be used to compensate for this loss.
- The slow evolution of the appearance of the tracked object due to changes in illumination, orientation and shape for deformable object.
- The search space, often due to its high number of dimensions, can be very large making the real-time objective challenging.

One way to cope with the problems of partial-occlusion, noisy scene images and changes of the properties of the object is to define and use a probabilistic framework. For instance, a probability value for the location of the object can be estimated using a distance measure between the scene image and the template image. Note that the choice of this distance measure is crucial for reliable, robust and efficient tracking. If the measure is applied at any point of the image, a probabilistic density functions for the object location is obtained.

The observation that, in most applications, object motion is tightly bounded, *i.e.* the object's positions do not differ much between following frames, can be used to achieve the real-time requirement. Moreover knowing the speed of an object or the characteristics of its movement can help to better predict the future object location.

The usage of statistical methods, namely Monte-Carlo methods, to estimate the location of an object has proved useful to reduce the search area. Note that since the matching has to be done in multiple locations, according to the algorithm used, parallelisation is likely to scale up well the speed of the tracking process.

In a class of algorithm the tracking process can essentially be divided into two sub-processes:

- A process to decide where to look. This restricts the search area and thus addresses the real-time objectives
- A process to decide how to look. This measure the probability of presence of an object.

One such technique that makes this distinction, and that we explore in more detail in this thesis, is the particle filter. Chapter 3 provides a detailed discussion based on the implementation of particle filters to track a translucent micro-pipette tip.

While particle filters are efficient the measurement method that were used were both inefficient and inaccurate. To remediate to this issue alternative techniques were explored. One such comparison technique that was considered is very much related with the Hough transform. It roughly consists of counting the number of features that match a model at a given position. Not only is the technique robust and suitable for microscope images that can be noisy, but it can also be implemented in such an efficient way that an exhaustive search of a bounded region of interest can be carried out in real-time. This is because for microscope manipulation the motion of object can generally be constrained to have 4 degrees of freedom: 3 translations and one rotation around an axis perpendicular to the field of view, resulting in a state space small enough to be explored in real-time with the presented technique.

This led to the main contribution of this thesis: template reduction of point feature models. A novel framework has been introduced in order to reduce the number of points a point feature model requires to robustly and efficiently identify a shape in an image. Since fewer points are required the matching can be done more efficiently. Recognition, identification and tracking algorithm efficiency can benefit from this improvement.

In order to perform this template reduction the state space of the object is considered: sets of points that characterise a unique state of the object across the whole state space are reduced template candidates. Constraints of robustness to missing feature points and additional feature points not corresponding to the object are also examined. A practical algorithm based on a generalised version of the Hough transform is presented to implement the template reduction.

1.3 Research methodologies

A major part of the work undertaken in this thesis has consisted of developing and implementing algorithms. Implementations have been made publicly available online as open source so that it may be publicly scrutinised. Much of it can be found integrated into the Mimas [5] library.

Mimas is a C++ vision system toolkit built in-house, at Sheffield Hallam university, with an emphasis on real-time applications. The library provides the infrastructure to load images from a wide range of formats due to the integration with other open source libraries such as the ImageMagick library, video by using the Xine library and cameras supporting Video for Linux and Firewire. Mimas also provides a number of low level image processing algorithms, such as edge/corner detection, disparity map, morphological operators or camera calibration. As well as higher level algorithms for object recognition and tracking such as geometric hashing and particle filters. Various optimisation schemes, such as POSIX threads, are implemented in order to achieve the real-time objectives.

1.4 Contributions

The main contributions of this thesis, and its associated chapters are:

- The development of a novel framework to reduce templates that are made-up of point features for rigid objects. The framework considers the state space of an object in order to determine a subset of features of the original template that characterises uniquely and robustly the state of the object across all the state space. An algorithm based on the Hough transform is used to select these features. An introduction to the problem and its formalisation is presented in chapter 4. Material presented in chapter 5 is closely related with the issue.

- A new shape estimator:the stencil estimator. It is shown how the estimator can be evaluated in an efficient manner using a variation of the Hough transform. This is presented in chapter 5 and an evaluation of the method can be found in section 6.2.

- Generic improvements to the particle filter algorithm in the imaging context. An algorithm is introduced to cluster particles. Subsequent analyses following the measurement process, that evaluate the posterior probability density function of the state of the object, is discussed. These analyses can provide feedback to the particle filter to partially reinitialise the sampling of the particles. This is found in sections 3.5 and 3.6.

- Application of the the stencil estimator in a microscope environment where objects are subject to changes in appearance. Microscopes have a narrow depth of field and the features of an object changes with its distance to the microscope's objective. This characteristic can be used to determine the distance of an object to the microscope's objective by using a model of the object that consists of a stack of images of the object taken at different depths. This contribution is presented in section 6.1.

1.5 Organisation of the thesis

Chapter 2 of this thesis presents a literature review focused on computer vision techniques related to the efficient identification and tracking of 2-D shapes in images.

The particle filtering algorithm is then presented on a concrete application, the tracking of a translucent pipette tip, and a few generic improvements are proposed in chapter 3.

In chapter 4 a novel approach to selecting sets of characterising points of a shape is presented. When these sets of points are identified in an image, they allow the state of a shape to be uniquely and robustly identified. The novelty consists of using the state space to select these characterising sets of points. This technique can be used during a pre-processing stage to obtain a more efficient representation of tracked objects.

In chapter 5 we present the stencilled Hough transform. The technique is a modification of the bounded Hough transform [6]. The introduction of the stencil estimator combined with a methodology to reduce the area of the stencils improves the speed and memory usage of the algorithm substantially, as shown by the test results presented in section 2 of chapter 6.

In chapter 6, experimental results are presented where the tracking of micro-objects under a microscope was performed with 4 degree of freedom using a variation of the bounded Hough transform. In other experiments the bounded Hough transform and the the particle filter were compared on the tracking of a translucent pipette tip. Advantages and limitations of both techniques are discussed.

Chapter 7 concludes this thesis, and future research directions are proposed.

Chapter 2

Literature review

2.1 Introduction

Identifying a shape in an image is one of the fundamental problems in computer and machine vision. The Hough transform and the wide variety of techniques that are derived from it are able to tackle this challenge to some extent. Other approaches include active contours [7] such as snakes [8] or level sets [9], matching techniques such as the inefficient cross-correlation coefficient [10] or boosted filters [11], and neural networks [12]. Each of them can outperform the others in specific application domains. While active contours can handle smooth deformable shapes that change, neural networks have been used extensively for hand-written character recognition. Boosted filters have been used for the recognition of object classes such as the human face whereas Hough transform techniques are well suited for rigid objects. Hough transform related techniques have been used, for many years now, to recognise 2-D objects in real-time on standard desktops. Although the shape of an object needs to be known, Hough transform related techniques do not need to be trained, are resistant to noise and partial occlusions and can accurately determine a specific object location.

Choosing the right technique for a given application, although essential, is not straightforward given the wide choice of techniques currently available. One of the major concerns, that has to be kept in mind and may help in this choice, is the real-time constraint. Indeed, the machine vision system should not be a bottleneck in the application that it is embedded into and, as a consequence, should be as fast as possible. A general way to speed up object location algorithms is to constrain the search area. Computational costs tend to increase dramatically with the size of the search area. To constrain the search area a number of techniques can be used. Nevertheless, they share a common trait: to utilise the object's previous location or, more accurately, its estimated location and some knowledge of the object motion.

These techniques are commonly referred to as tracking techniques. From a utilitarian point of view tracking has to deal with a number of issues that have been presented in section 1.2.1, page 5, and will be further examined on a practical application in section 3.3, page 40. A few different approaches to tracking will be discussed in the visual tracking section of this chapter. One group of approaches consists in distinguishing a component that selects the locations where to look for the object from a component that identifies the object. When this distinction is possible, tracking often refers to the first stage of these approaches.

Nevertheless to be able to track an object its initial position has to be determined, the review is started by presenting the initialisation stage, this stage can make use of an approach that is often overlooked: motion detection, and when the background remains stable background subtraction.

In this thesis, we deal mostly with template matching as the identifying method, *i.e.* the usage of an image of the object to identify its characteristics in another image which is usually larger and that contains other elements. However other methods exists, for instance, the usage of parameter or generative models [13] could be used. Moreover, the focus has been put on matching techniques that are feature based. The underlying assumption being that by reducing the data from colour images to a set of features, the data may be more efficiently utilised. This is arguable and discussed in Zitová and Flusser [14], who provide a survey on image registration techniques. The efficiency of determining the characteristics of an object in an image depends on:

- how discriminative the features are *i.e.* how many of them are needed to determine the characteristics of an object.
- the cost of evaluation of these features.
- how features are compared and used to determine the object characteristics.

Since the solutions to each of these issues are interdependent and also depend on a variety of parameters on the problem under consideration it is an open and complex topic of research. Some elements that have to be considered will be discussed in this thesis. More specifically, the issue of reducing an object representation that consists of feature points while keeping a discriminative object representation will be tackled.

Another approach that has been the focus of research for the matching of templates is the hit-or-miss method which emerged from the field of mathematical morphology. A review of this approach is provided in this chapter.

There are many other alternatives to characterise, represent and analyse shapes, Loncaric [15], in a not so recent survey, presented a number of these techniques. This survey can serve as an introduction to shape analysis. Walker, Cootes and Taylor [16] proposed another interesting alternative for 2-D object representation: the use of salient features to track an object in a video sequence and automatically build its appearance model [17][18]. As previously mentioned our focus is on feature points representation, techniques to match point sets are reviewed in the section justly named: techniques to match point sets.

Distortions can be quite common when working with custom cameras and microscopes. A few issues related to image distortions as well as how to handle them are presented.

The main results of this thesis are related to the Hough transform, hence after an introduction presenting a generalised Hough transform the link with tracking is established and more advanced topics, which are related to the work, are then discussed in the last section. They deal with improvements to the original Hough transform algorithm.

2.2 Connected techniques to visual tracking

2.2.1 Detection and initialisation Tracking essentially makes use of the previous position of the object to reduce the search space where the object is searched for in the next frame. However, the object has to be located in the first frame. Moreover, when the tracking fails, a recovery system needs to be used to locate the object again. For these two reasons, it is practical to use a tracking algorithm in combination with a detection algorithm.

As detection techniques become more efficient they supplant tracking methods in some applications since it is possible to track object in an image without any assumption on its probable location. Detection algorithms may be simpler to implement, but they come with a cost penalty in terms of efficiency due to the fact that important informations such as the previous location of the object are discarded. As usual, the choice depends on the resources available (computational power, time, *etc.*) and the application under consideration.

Nevertheless, when multiple instances of the same object are present, such as cars on a motorway or people in underground transport, tracking cannot be replaced entirely by detection when the object identity needs to be known.

2.2.2 Background subtraction and movement detection For the problem of initialisation a very useful cue is the object's motion. If the background is not moving or if its appearance is stable (of uniform colour, for instance) then object features can be selected by subtracting features in the background. A review of background subtraction techniques is provided by Piccardi [19].

Sugrue and Davies [20] developed an original approach which consists of applying a 3-D spatio-temporal filter on a stack of sequential images to detect motion. Its advantages over background modelling techniques are that no information from the background is needed, which makes it faster. The technique is also inherently robust to noise.

2.2.3 Mathematical Morphology Ronse, Naegel and Passat [21][22][23] presented a very impressive and elegant theoretical framework by combining multiple research strands on template matching using the tools of mathematical morphology and their generalisation for grey-level images. The algebraic approach has its limitations though, it focuses more on the exactitude of the methodology rather than performance evaluation and implementation issues.

A characteristic of morphological operators is that they are not forgiving; when a shape of a binary image differs by a single point from the structuring elements then the whole "shape" is discarded. Although the hit-or-miss transform for grey level images provides some tolerance to additive Gaussian noise, it is not very robust to perturbations. To cope with this issue, rank order filters have been designed [24]. Also the notion of fuzzy sets has been used to further improve robustness as discussed by Gasteratos and Andreadis [25]. This idea of a more tolerant framework using fuzzy morphology is also discussed in [26].

The literature is very scarce in algorithm implementation and performance comparisons. Using the openCV library a few tests were run. For a colour image of 512×512 (~ 250000) pixels on an Intel Core 2 Duo (T5450) 1.66 GHz (~ 3300 bogomips) the approximated processing times are given in the following array:

Diameter of the structuring element	3	5
Opening or closing operator	180 ms	230ms
Erode or dilate operator	95 ms	140 ms

It can be seen that morphological operators can be time consuming. To circumvent this issue different approaches have been taken: using field-programmable gate arrays (*FPGA*), using optical processors, but the most promising approach might be, because of its future availability and sheer processing power, to make use of *GPGPU*s (General-Purpose computation on Graphical Processor Unit) such as *CUDA* (Compute Unified Device Architecture) enabled *GPU*s.

One of the main advantage of expressing template matching in terms of morphological operators is that they can be implemented on hardware like *FPGA*. Baumann and Tinembart [27] provide an overview of the tools and insights in the methodology that could be used to implement morphological operators on a *FPGA*. A hardware implementation is also discussed in [25].

The literature also contains references to an optical implementation [28], *i.e.* by using an optical processor however this is beyond the scope of this review.

A notable issue with morphological operators is that they are translation invariant operators relative to a structuring elements. In [23], which presents an application for the segmentation of blood vessels from 3-D angiographic[1] data, this issue is circumvented by rotating the structuring elements.

2.2.4 Techniques to match point sets The selection of a measure to identify an object depends on its properties and its motion. Different techniques may be more or less suitable and efficient when an object is deformable and moving in a 3-D

[1]The examination of the blood vessels using X-rays following the injection of a radio-opaque substance.

space, or rigid and undergoing 2-D translations. If practical results are expected, identifying these characteristics is critical for the selection of an efficient technique.

When distortions, due to an imperfect lens can be neglected, the transformations that a 2-D plane undergoes can be modelled by projective transformations. Moreover, when the object is small compared with its distance to the camera and if it moves by a small distance according to the same criteria, perspective effects can be neglected and the object features can be put into correspondence using affine transformations. An affine transformation of the plane consists of a linear transformation followed by a translation. An affine transformation can be considered as a composition of a dilation, a rotation, a shear and a translation. Note that lengths and angles do not remain the same under affine transformations. The affine transformation subset that have these properties is called the similarity group; it consists of the Euclidean transformations and the mirror transformation.

Gope and Kehtarnavaz [29] provided a review of affine invariant comparison techniques of point sets. They also described a technique where an image point set and a template point set are compared relatively to an affine transformation. In order to do that the parameters of the affine transformation are evaluated by determining affine invariant points for both sets of points and comparing them. Then the match is validated using an enhanced Hausdorff distance method. Finally, the technique is compared, using a dataset, to three popular affine invariant techniques. The outcome is favourable for the presented technique and dataset in terms of efficiency, noise and occlusion resistance. The Hausdorff distance between two sets of points U and V is defined as:

$$\mathrm{H}(U, V) = \max(\mathrm{h}(U, V), \mathrm{h}(V, U)) \tag{2.1}$$

where

$$\mathrm{h}(U, V) = \max_{u \in U} \min_{v \in V} \|u, v\| \tag{2.2}$$

In order to cope with occlusions and outliers different schemes have been used instead. For instance instead of using the maximum, the k^{th} ranked distance can be used. The paper proposes an interesting variation of the Hausdorff distance. For more details on the Hausdorff distance and its implementation the reader may also refer to [30].

The techniques above assume that individual features are not distinguishable.

However the measure could be adapted by taking into account an additional criteria, such as colour for instance, conditioning the evaluation of the distance of points belonging to the two sets U and V. Combining global geometric characteristics with local appearance characteristics of an object is an idea that can greatly improve reliability and efficiency of many measures. Many interesting studies examining the synergy of these two fundamental aspects of object identification could be done.

A useful mathematical structure to represent and manipulate transformations of the general affine group, GA(2), or the projective transformations in $\mathbb{R}^2$, where P(2), is the Lie algebra [31][32][33]. It allows the separation of different components of a transformation in terms of, for instance, translation along the x and y axes, rotation, dilation, shear and shear at 45 degrees. This is valuable when these information are needed by a system like a robotic arm [33].

Another useful technique to compute corresponding sets of points is the Chamfer distance. It consists in evaluating the distance of a pixel to its closest feature points. Such a distance map can be evaluated efficiently [34]. It allows the usage of optimisation algorithms using gradient techniques such as Levenberg-Marquardt to minimise an energy function that characterises the matching of two sets of points.

In Robust Registration of 2-D and 3-D Point Sets [32] Fitzgibbon has compared the iterative closest point (*ICP*) algorithm and a Levenberg-Marquardt based algorithm. These techniques require a previous approximation of the location of the object and are therefore well suited for tracking when the object position does not change significantly from one frame to another.

Breuel [35][36] presented a technique to match geometric primitives, such as points, given a transformation space. The techniques based on branch-and-bound methods finds a global optimal solution to the matching problem. It works by recursively subdividing the transformation space and computing the upper bound of the number of points that can be potentially mapped by a transformation of a sub-transformation space. This is referred to as the matching quality of the sub-region and denoted by $Q(\mathbb{T})$. The computation of the bound of $Q(\mathbb{T})$ for a transformation space $\mathbb{T}$ is easy, that is why the technique works well, for more details on this point refer to the paper. The technique is exhaustive and guarantees a globally optimal solution to the geometric matching problems.

The algorithm works as follows:

1. The algorithm maintains a priority queue of search states. When two search states have the same priority, the state with the higher depth in the search tree is preferred. The queue is initialised with a state representing all possible solutions.

2. Each search state $\mathbb{T}_k$ is associated with a sub-region of the transformation space $\mathbb{T}_k \subseteq \mathbb{T}_{all}$. It is further associated with an upper bound such that $\forall T \in \mathbb{T}_k : \hat{Q}_k = \hat{Q}(\mathbb{T}_k) \geq Q(T)$; the upper bound serves as the priority of the state. For termination and correctness, the upper bound needs to satisfy that the sub-region is small enough and that its quality still remains above a given pre-determined threshold.

3. The algorithm removes the state with the highest upper bound from the priority queue. In case of ties, states with higher depth in the search tree are preferred. Breuel showed that depth first search is almost as fast as breadth first and uses only a tiny fraction of memory.

4. The transformation $T \in \mathbb{T}_k$; if $Q(T) = \hat{Q}(T_k)$, terminates the search and returns T as a solution.

5. Otherwise, the region $\mathbb{T}_k$ is split into two disjoint sub-regions $\mathbb{T}_{2k}$ and $\mathbb{T}_{2k+1}$ such that $\mathbb{T}_k = \mathbb{T}_{2k} \cup \mathbb{T}_{2k+1}$ along its largest dimension. $Q(T_{2k})$ and $Q(T_{2k+1})$ are evaluated and these sub-regions are inserted back into the priority queue.

Geometric hashing is another technique to estimate an object position using feature points. It consists of selecting group points from the template image that would characterise the location of the object if they could be matched in the image. For instance, if translation and rotation on the plane of an object are considered, the correspondence of any group of 2 points between the template and the image features is sufficient to determine the object position. Each of this group of points can be characterised by the geometric distribution of the remaining feature points. For instance, expanding on our example, a group of two points determines an axis that can be used to position a grid on the template image. By using the number of features belonging to the respective elements of the grid the group of two points can be characterised by a signature. In the image, a group of potentially characterising points is randomly selected, its signature is evaluated and compared with the signature of the template group of points. When a match is found the potential position of the object is evaluated. The operation is iterated until the time allocated has expired or enough evidence has been accumulated to estimate the object position beyond reasonable doubt. Our experiments have shown that, for the example we mentioned, the first match is sufficient. Details of our implementation is available in Wedekind *et al.* [37]. The method has been extended to recognise objects using focus stacks, allowing the distance of the object from the lens of a microscope to be estimated.

As mentioned in the introduction there are other alternatives for identifying a shape. One of these alternatives is the use of moments. Moments [38] can be efficiently evaluated [39] and can provide a compact representation of a shape by characterising its global features. For instance, using the moments up to the second degree a shape can be approximated by an ellipse. Many applications in image analysis have been found for them: object classification, pose estimation, pattern recognition and compression. However, the global characterisation of a shape is not robust to occlusion which reduces its application domain to specific but nevertheless useful environments, for visual servoing for instance. Even though moments can not be used to characterise the global shape of an object in case of occlusion, by characterising local patches, for instance by using the orthogonal and rotation invariant Zernike moments [40], the global pose of an object can still be efficiently and robustly determined. Thus moments can be used to characterise features. Having distinguishable features redefines the problem of matching sets of feature points and reduces drastically its complexity [41]. In appendix B.2 other feature characterising methods are mentioned.

2.2.5 Image distortion It is not uncommon that the lens system of microscope produces image distortions. To correct them, so that the shape of an object remains the same independently of its position within an image, the image acquisition system has to be modelled. In most cases, distortions exhibit a central symmetry on the principal point of the image that can be corrected by an adequate mapping of the image.

The mapping of the world - the geometry of which can be modelled with a 3-D vector space - to a 2-D discrete space, the image, has already been modelled successfully through various methods [42]. According to the chosen model, a slightly different set of parameters have to be found. These parameters depends on the characteristics of each camera. The process of determining these parameters is known as camera calibration.

Camera calibration is still an active field of research, although the technology is mature enough so that different implementations are freely available. Such implementations can be found for instance in the Mimas, OpenCV and Gandalf libraries. Thus, the camera calibration problem boils down to identifying a suitable model for our requirements and an implementation to obtain the model's parameters.

Camera parameters are generally classified into two classes: the intrinsic and the extrinsic parameters of the camera. The intrinsic parameters are the focal length, the coordinates of the principal point and a few parameters to model the geometric distortions characteristic of the lens system. The extrinsic parameters are the position of the camera, *i.e.* location and orientation compared to an arbitrary external frame.

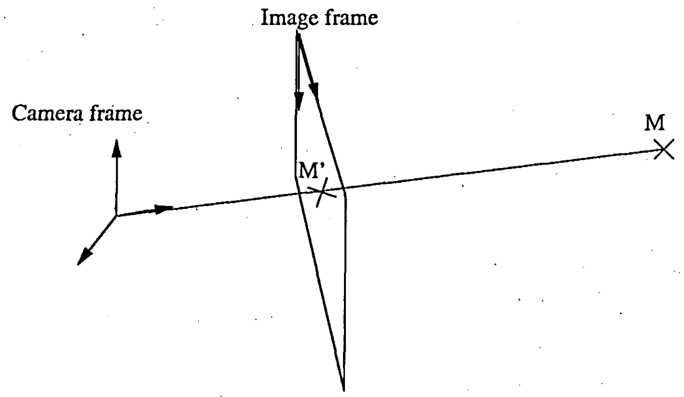


Figure 2.1: By obtaining the intrinsic parameter it is possible to determine where each point expressed in the camera frame will be projected onto the image.

The process of calibration needs to be performed only once and thus can be conducted off-line. The calibration process involves taking images of a scene where 3-D points of the scene are known. By finding the correspondence points in the resulting images, the parameters are found by solving a system of equations. Once the camera is calibrated it is possible to associate a 3-D ray to each pixel of the image as illustrated in figure 2.1 or to predict the 2-D location in an image of a 3-D point of the scene.

We have tested a method that uses a calibration object [42]. This object is a grid similar to a checker board with known measures. The calibration grid is shown in figure B.5 on page 176, in the appendix. Once the intrinsic parameters of the camera are known it is possible to project a model of an object in the image as illustrated in Figure 2.2. Using the intrinsic parameters of the camera we have mapped a cuboid having the dimensions of the chess board with the chess board in the image. The mapping was performed manually by trials to estimate the location and rotation of the chess board.

This process can be automated if there exists a way to estimate the pose of the object. This is actually the basis of model-based 3-D pose estimation which was one track that was considered to pose estimate microscope objects. In appendix B.1, stereo-vision is discussed in more depth, however for microscope manipulation the physical space available to position the image acquisition system is limited making it complex to have multiple image acquisition systems. Different systems of mirrors were also considered but were judged too complex to be implemented. The results of some experiments made with a Rubik's cube are presented in section C.2 of this thesis. For 3-D pose estimation of an object the conclusion that a 3-D representation of an object is needed was reached, in appendix B.2 a few alternatives are considered to this purpose.

In figure 2.2, the white line at the top of the image is curved due to lens deformation. The line is mapped on the image to the approximated, manually estimated, position of an edge of the power supply unit at the top left corner of the image. It demonstrates that the deformations of the image by the lens system are taken into account by the model when a model is re-projected onto the image.

Distortion parameters can also be used to rectify an image. A rectified image is an image such that lines into the image are projections of lines from the real world. This is illustrated by figure 2.3.

2.3 Visual tracking

Amongst the various visual tracking algorithms, one class of algorithms divides the tracking problem into two sub-sections: where to look for the object and how to look for it. Kalman filtering and particle filtering belongs to that class of algorithms. They are first reviewed. Nevertheless, this division is not always possible. For instance, energy minimising methods are designed to perform both operations simultaneously. Some examples of this category of algorithms are subsequently reviewed.

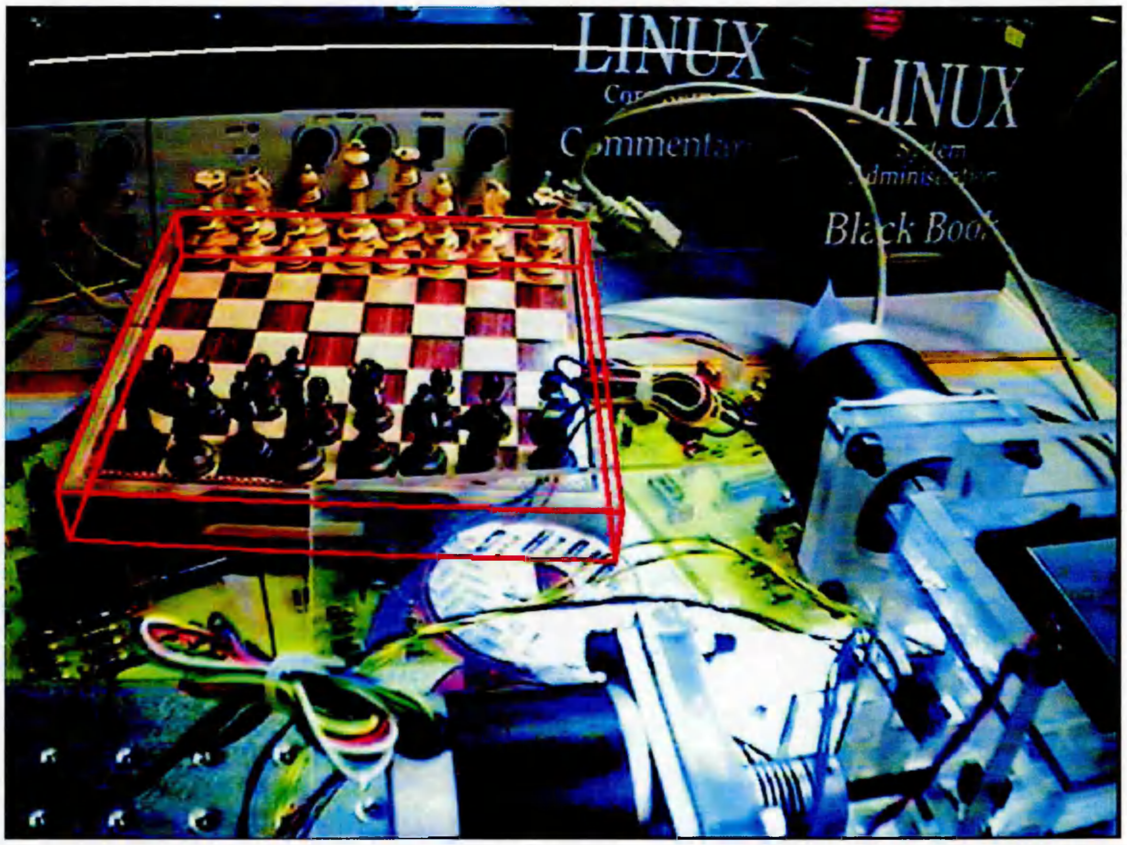


Figure 2.2: A red parallelepiped and a white line projected on the image using the image formation model

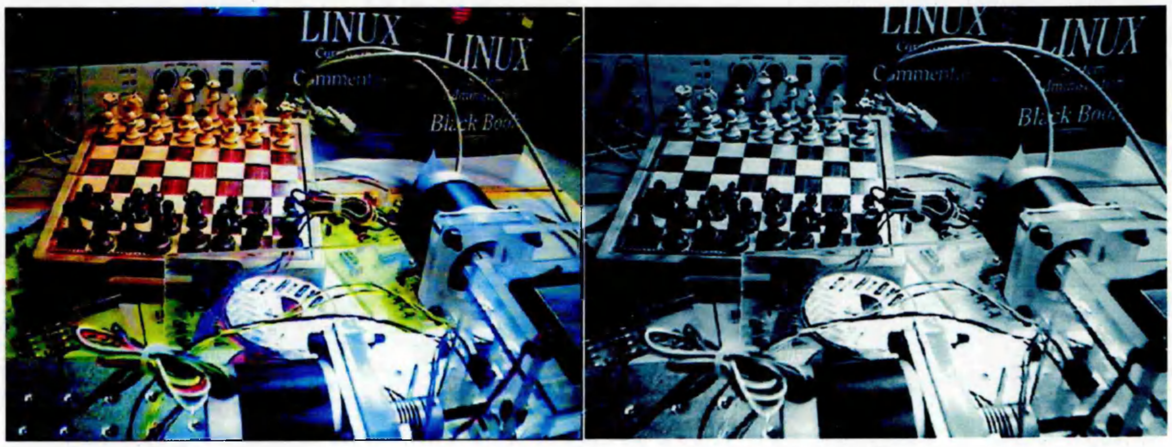


Figure 2.3: The right image is the rectified left image. Notice that the power supply edge highlighted in previous figure appears straight.

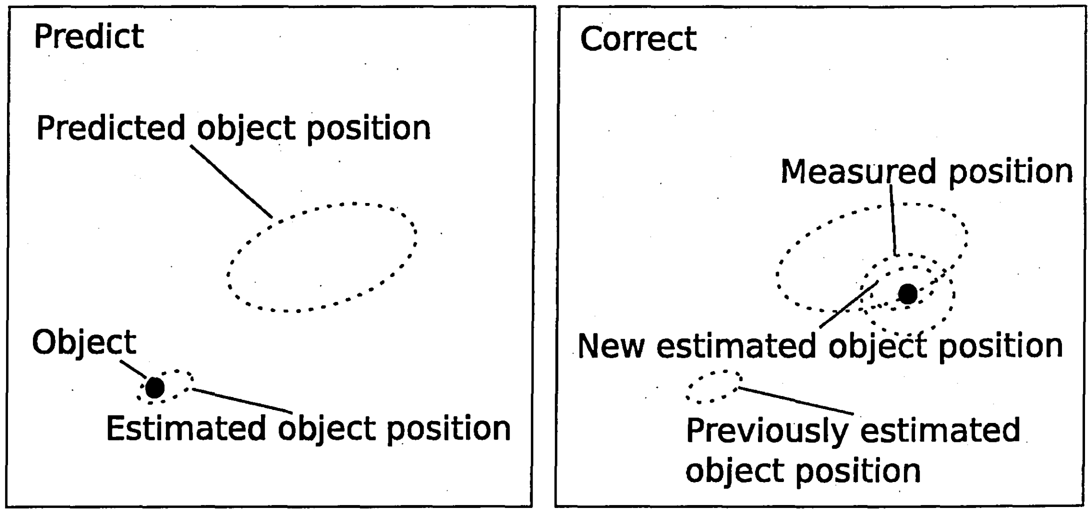

Predict
Predicted object position
Object
Estimated object position
Correct
Measured position
New estimated object position
Previously estimated object position

Figure 2.4: The Kalman filter can be divided into 2 stages

2.3.1 Kalman filter based visual tracking In the Kalman filter the hypothesised state of the tracked object is described by a random vector assumed to follow a Gaussian distribution characterised by its most probable state and its covariance matrix. Additionally, the evolution of the system is assumed to be modelled by a system of linear difference equations. Kalman filtering integrates the information from the previous estimated states of the tracked object, through the motion model of the system, and the new information provided by the image. The previous estimated states of the object determines the search area in the image where additional information is collected to more accurately evaluate the pose of the object. According to the reliability of the measurements, characterised by its covariance matrix, the confidence on the new data and the previous collected data is weighted to obtain a new estimate of the state of the object. The numerical determination of these weights minimises the expectancy of the error covariance, which could be interpreted as the uncertainty of the state of the object. Welch and Bishop [43] provide an introduction with the mathematical derivation of the technique and a simple example on which the Kalman filter is applied. Figure 2.4 illustrates this process on a simple example where the x-y position of a black sphere is being tracked. The dotted ellipses correspond to the one-σ distance from the expected value of the variables indicated by the captions.

At implementation time one usual issue is to evaluate the uncertainty of measures, however different case studies, experiments and simulations have shown that state estimation tend to remain robust even with an approximative evaluation of the uncertainty of the object state. This robustness leads to the successful estimation of the state of an object in spite of the imperfect knowledge of the measurement uncertainty that happens in practice.

The extended Kalman filter was designed in order to deal with systems which behaviour cannot be modelled by a system of *linear* stochastic difference equation. To solve the system of equations, the system is linearised using a Taylor expansion about the two first moments of the random vector representing the state of the system. However, when the motion model is not linear, for visual tracking, the unscented Kalman filter [44] is almost always preferred.

In the unscented Kalman filter [44], by using the previous estimated state of the object a number of states depending on the size of the random vector representing the state of the tracked object are determined. This limited number of states, called sigma points, and that are distributed around the mean of the Gaussian distribution are used with the motion model to determine the locations where the measurements are taken in the subsequent image. Compared with the extended Kalman filter the implementation is facilitated as there is no need to evaluate any Jacobian matrix; also the unscented Kalman filter deals better with highly non linear functions since no linearisation is made and the search space is better determined thus more efficiently capturing the probability distribution of the state of the object. They are similitudes with particle filtering however there exists two main differences: the distribution is assumed to be Gaussian and its sampling is structured as opposed to random.

Many references – news items, papers, source code and applications – about the Kalman fitler can be found on the webpage [45] maintained by Greg Welch and Gary Bishop.

Two examples of application are now provided. Stenger *et al.* [46] performs hand tracking using quadrics to model a hand and the unscented Kalman filter, notice that although the hand model has 27 degrees of freedom the tracking is only demonstrated for 3-D rigid movements of the hand. Youngrock Yoon in his thesis [47] uses the extended Kalman filter to track in real-time with their 6 degrees of freedom rigid objects. The object model used corresponds to the edges of the object, the process to obtain these model from range images is thoroughly described. This work is quite characteristic of the hidden complexity of practical details: extracting edges from the image, discretisation issues with lines, self-occlusion of some edges by the object, only partial extraction of all edges which are usually truncated and the list goes on. Developing schemes to work around these issues is complex, not merely the case of applying a well described algorithm such as the Kalman filter or its extensions. This shows an interesting contrast with the neat and clear description provided in the previously cited paper [46] which also masks completely all practical issues.

2.3.2 Particle filter based visual tracking Particle filtering is a robust, versatile, real-time[2] component of a system for the visual tracking of objects in video sequences. Particle filters have been very popular in the last few years and a large number of publications have reported adaptations of particle filters with various image metrics [48][49].

A number of tutorials [50], reviews and historical accounts on particle filters [51][52][53] are available. Only a brief summary of the most important points is given in this thesis. Mathematical derivations [54] are not presented here, however, some of the current notation and vocabulary associated with particle filters are introduced. Particle filtering is a sequential Monte Carlo methodology that uses a Bayesian framework to predict the future probable location of an object and sample the state space accordingly. This method is sequential or iterative because the information arrives in sequence, image after image. Monte Carlo methods are used when the state space is too large to be explored exhaustively, typically due to a high number of dimensions, the state space is sampled in order to extract meaningful information and, in the case of tracking, to approximate the probability density function of the object state. It uses a Bayesian framework because Bayes' theorem is used to infer the probability density function of the location of the tracked object knowing the measures taken on the current image.

As mentioned in the previous section, Kalman filtering [43] as well as other approaches where probabilistic density functions are assumed to be unimodal Gaussian, works relatively poorly in the presence of cluttered backgrounds. *CONDENSATION*

[2]Particle filters alone cannot track an object or a shape, they have to be associated with a measure. Since the time consuming process is the measurement process, the realisation of a real-time implementations depends largely on the choice of this process. The infrastructure to implement particle filters does not require a huge amount of space or computation compared with the capabilities of a standard computer and the requirement in number of particles of efficient measurement methods.

(CONDitional DENSity propagATION) [54], which is a particular instance of particle filtering, is an algorithm developed to solve the above problem. The *condensation* algorithm is capable of supporting multi-modal distributions. It is based on sampling the posterior distribution estimated in the previous frame and propagating these samples or particles to form the prior distribution for the current frame. However, it requires a relatively large number of samples to ensure a fair maximum likelihood estimate of the current state and can be computationally more expensive than the unscented Kalman filter for instance.

The different properties (*e.g.* position, shape, size, configuration) of a tracked object are described in the time-stamped state vector X_t while the vector Z_t denotes all the observations, which are images, $\{z_1 \dots z_t\}$ up to time t. Using the data obtained by a sensor device, X_t can only be known to a certain degree that is why probability density functions (*pdf*) are used to model the tracked object characteristics. Compared with Kalman filters which assume Gaussianity and unimodality of the *pdf*, particle filters do not make any functional assumption on the shape of the *pdf*. This is a major advantage since in practice, due to the presence of clutter, posterior density $p(X_t|Z_t)$ is often a multi-modal *pdf*. However, when the posterior density is unimodal there is an unnecessary computational cost due to unnecessary additional measurements.

The key idea of particle filtering is to approximate the probability distribution of the current state by a weighted sample set: $S = \{(s^{(n)}, \pi^{(n)})|n = 1 \dots N\}$. Each sample consists of an element s which represents the hypothetical state of an object and a corresponding discrete sampling probability π where $\sum_{n=1}^{N} \pi^{(n)} = 1$. The evolution of the sample set is described by propagating each sample according to a motion model as follows: each element of the set is weighted by applying a measure to the last captured image, $\hat{\pi}^{(n)} = p(z_t|X_t = s_t^{(n)})$, and normalising the weights $\pi^{(n)} = \frac{\hat{\pi}^{(n)}}{\sum_i \hat{\pi}^{(i)}}$; $p(z_t|X_t)$ is known as the observation or measurement density. The prior for the next frame, also referred to as the prediction density, $p(X_{t+1}|Z_t)$ is then evaluated by randomly drawing the samples $\{(s^{(i)}, \pi^{(i)})\}_i$, with replacement, according to their weight and by applying to them the motion model $p(X_{t+1}|X_t)$. Moments, such as the mean of the samples, might then be evaluated to estimate the state of the object:

$$E[S] = \sum_{n=1}^{N} \pi^{(n)} s^{(n)} \tag{2.3}$$

Thus particle filters are able to consider multiple state hypotheses simultaneously. Moreover, since less likely object states have a chance to temporarily remain in the tracking process, particle filters can deal with short-lived occlusions.

Particle filtering can be used for tracking people, cars or faces in cluttered environment where multiple instances of a class of objects can be present. In Nummiaro *et al.* [48] particle filters are combined with colour histograms. The Battacharyya distance measure, which is described in the section below, was used to evaluate the validity of the hypothesised locations. A methodology to adapt the object representation, *i.e.* the colour histogram of the template, to cope with the slow variation of the appearance of the object has also been proposed by the authors: a mixture of the previous model and the current model is used as an updated template when the probability of tracking the correct model is high enough. This avoids updating the model when the target is occluded or when the image is too noisy. Using the previous notations, with an additional time index in superscript this process is given by: $\hat{p}_i^t = (1-\alpha)\hat{q}_i^t + \alpha\hat{p}_i^{t-1}$, $\alpha \in [0,1]$ where $\hat{\text{q}}$ represents the colour distribution of the estimated object position and $\hat{\text{p}}$ the colour distribution of the tracked model. Moreover, the performance of the colour-based particle filter is compared with the mean shift tracker [55] based on the Battacharyya coefficient and the mean shift tracker associated with a Kalman filter that predicts the likely position of the object in order to reduce the number of iterations needed by the mean shift algorithm when motion is fast. Computational performances are comparable and suitable for real-time tracking. Particle filtering is more consistent in the time needed to evaluate the position of an object. This is because the number of operations depends on the number of particles used to locate the object which remains constant whereas, for the mean shift algorithm, the amount of operations to be carried out can vary depending on how fast the algorithm converges. Particle and Kalman filtering have the additional advantage of taking scale changes into account. Particle filtering is more robust than the two other methods but locates the object less precisely. This work could be extended by combining the tracking of different parts of an object. For instance, a face tracker could be designed by combining the tracking of the mouth and the eyes and checking the coherence of the results of the different trackers.

2.3.3 Other visual tracking algorithms Comaniciu *et al.* [56] presents a gradient optimisation approach to track objects. The method uses a metric derived from the Bhattacharyya coefficient as the similarity measure which allows the tracking of deformable objects; as a result it behaves well during perspective transformations. For the given examples the method coped well with noise, partial occlusions and clutter. The emphasis is put on the target representation. The distance between 2 discrete distributions is defined as:

$$d(\hat{\mathrm{p}}, \hat{\mathrm{q}}) = \sqrt{1 - \rho(\hat{\mathrm{p}}, \hat{\mathrm{q}})} \tag{2.4}$$

where

$$\hat{\mathrm{p}} = \{\hat{p}_i\}_{i=0}^{m} \qquad \sum_{i=0}^{m} \hat{p}_i = 1$$

$$\hat{\mathrm{q}} = \{\hat{q}_i\}_{i=0}^{m} \qquad \sum_{i=0}^{m} \hat{q}_i = 1 \tag{2.5}$$

are the m-binned colour distributions of the model and the target. And

$$\rho(\hat{\mathrm{p}}, \hat{\mathrm{q}}) = \sum_{i=0}^{m} \sqrt{p_i q_i} \tag{2.6}$$

is the sample estimate of the Battacharyya coefficient. The usage of an isotropic kernel on the spatial domain allows the distance measure to be differentiable in the neighbourhood of the object position and gradient based optimisation to be used to determine the best match instead of an expensive exhaustive search.

Collins [57] presented a technique using blobs and the mean shift algorithm [55] to track a human body. A BLOB (Binary Large OBject) is an area of connected pixels with the same logical state, for instance, a group of contiguous pixels having a colour compatible with human skin and being large enough. Lindeberg's theory of feature scale selection [58][59] has been used to adapt the area size of the kernel where a measure to evaluate the blob shift is taken into account. Changes of the blob area size are thus taken into account. Collins also explains how to adapt the mean shift algorithm to take into account negative weights which is useful when the measure, which is used to determine whether a pixel belongs to the foreground (the blob) or the background, can be negative. The mean shift algorithm evaluates the position of an object by iteratively computing an offset Δ_x from the current location to the next location using the following formula:

$$\Delta_x = \frac{\sum_a K(a-x)w(a)(a-x)}{\sum_a |K(a-x)w(a)|}$$

where x is the current location, K is a kernel function and w a function that evaluates the likelihood of the pixel to belong to the object.

An overview of different techniques to track rigid objects using a single camera is given in [60]. Amongst these techniques the Kanade-Lucas-Tomasi [61][62] tracker is of special interest as it allows tracking to be performed in real-time. It also uses a minimisation technique based on the Newton-Raphson optimisation algorithm where the sum of squared intensity differences, also referred as SSD matching, over a small window, typically 15 to 30 pixels, is used as the measurement method. The selection of features is also discussed. In this case features are square patches of images. The gradient over an image patch is evaluated and if its 2 eigenvalues are large enough, the image patch is considered to be a good feature. This leads to a well conditioned matrix for solving the feature tracking equations. Not only does it avoid the aperture problem[3] but it also makes the feature patch robust to noise. While tracking the features for an extended period of time, their appearance changes due to perspective deformation and non Lambertian surfaces of objects. Moreover some "good" features may not correspond to any physical point on an object: the patch may contain a foreground and a background object at the same time, and, occlusion should be detected for the tracking to be meaningful. Shi and Tomasi [63] proposed the use of an affine transformation model that allows these features to be detected and marked as outliers. This could be beneficial for a subsequent processing stage, *e.g.* shape from motion. Jin *et al.* [64] extended this work to be able to cope with illumination changes. A fast (20 frames per second, 1000 features on 1024×768 video) GPU-based implementation was developed by Sinha *et al.* [65].

There exists a wide variety of tracking techniques adapted for specific environments, and due to their inherent differences no comparative study has been established. Since there is no dominant technique, no steadfast rules exist to select an adequate tracker. Nevertheless, a literature review presenting the best trackers with their specificity would be useful as it would provide some guidance in selecting a

[3]The aperture problem occurs when the feature patch cannot be used to characterise completely its movement. For instance, along an edge, when the feature patch is small enough, displacement in the edge direction cannot be detected .

tracker according to the requirement of an application.

Moreover tracking cannot be considered independently of a certain number of connected problems: as we have seen, the choice of a tracking technique is tightly related to the choice of representation of an object. Object matching, object recognition and detection, object representation and pose estimation are other related issues to tracking. Some aspects of these issues are discussed in the remainder of this chapter.

2.4 The Hough transform

2.4.1 Introduction The Hough transform is a technique that has been used extensively to detect and recognise geometric patterns in images. Its advantages include robustness to noise and partial occlusion, the capacity to handle multiple occurrences as well as the possibility to be parallelised. However the method suffers from excessive storage requirement and computational complexity [66][67].

Many descriptions of the Hough transform in its basic form are available. It has been extensively used for the detection of lines, for which it even has an FPGA implementation [68].

Using figure 2.5 a generalised Hough transform is described. As will be apparent in the next section, there are many variations of the Hough transform. This version is said to be generalised because it works for any kind of shape and not only parametrised shapes such as lines or circles. In figure 2.5 the location of the " bust" shape that can be translated in the image space is considered. Therefore the parameter space consists of the x and y location of a preliminary chosen and arbitrary point of the "bust" shape. To locate the shape a point on the image space is selected and assumed to belong to the shape. Using this assumption the possible positions of the shape are calculated and voted for. This is illustrated in the figure by colours: the green point votes for the positions in the parameter space that are in the same colour. The dashed light green arrow symbolises this process. After repeating this process a certain number of times, the parameter that is most voted for in the parameter space is considered to describe the object position. In figure 2.5 only 6 points voted and the parameter that has been most voted for at that stage, circled in red, happens to represent the position of the shape. Note that another parameter is voted for 3 times, at another intersection of the violet, yellow and black

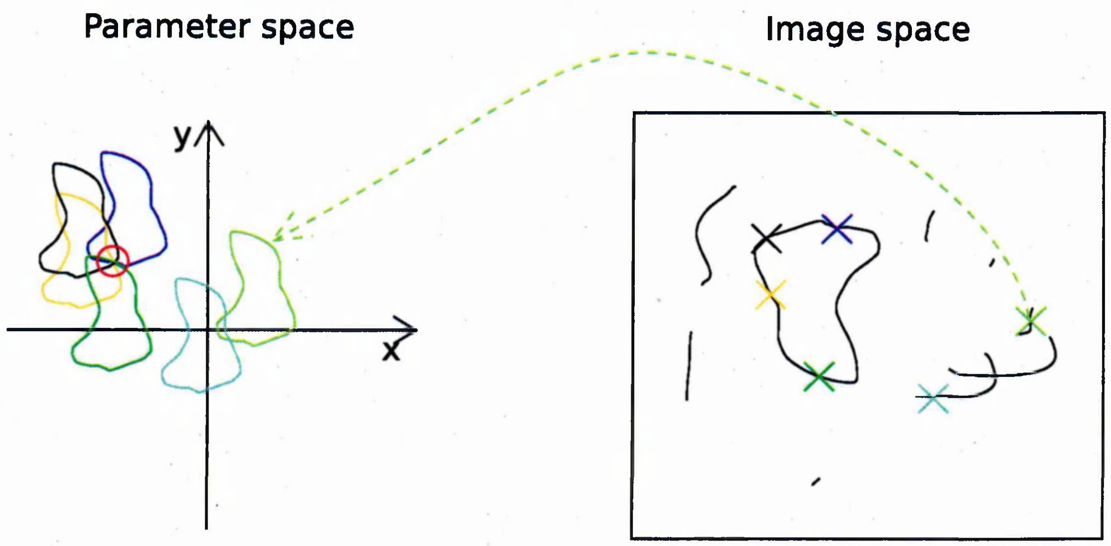


Figure 2.5: Diagram to explain the Hough transform, see text for explanation.

shapes, it would therefore be premature to conclude the position of the shape at that stage. As can be seen all sorts of variations can and have been adopted to make the Hough transform more robust, efficient and general; some of them will be discussed in more detail in the next section.

2.4.2 Hough transform and tracking

Most recognition and detection algorithms can be adapted to become tracking algorithms by focusing on a region of interest determined by the previous object locations. Within the context of this approach, the implementation of the Hough transform can be reorganised to be yet more efficient.

Except for the work by Greenspan *et al.* [6], we are unable to identify any other work where the Hough transform has been used for tracking in a similar way to the method presented in this thesis, possibly, because tracking involves a slight change of perspective. When using the Hough transform, the algorithm is implicitly set to be in a "writing" paradigm [69], *i.e.* for each image feature (say edge point) the accumulators of a parameter space are increased according to a reference table [70]. When performing recognition this factor is important because the parameter space is generally too vast to be handled rapidly whereas it is possible to consider relatively fewer image features. However, if the parameter space is kept small, which is possible when performing tracking, the unknown parameters can be determined in a different way: each possible transformation can be considered and the feature locations according to the transformation can be pre-calculated. This is equivalent to a "reading" paradigm, which might not be a sensible approach for detection when the parameter space is large.

These two paradigms are presented in [69] where it is shown how the Hough transform is related to the Radon transform, a different formalisation that generalises the Hough transform and allows it to be applied to a wider variety of mathematical objects such as continuous functions. This dual aspect of the Hough transform has been presented by many researchers [71][72] but did not seem to have been linked with tracking.

The essence of the tracking technique introduced in chapter 5 resides in the fact that the transformation space can be limited and that the feature positions can be pre-calculated and stored for all transformations. The transformation space is limited to the relative moves an object can make between two frames (say small translations or rotations).

In this thesis the object is modelled by a set of feature points. This approach is simpler than the more widespread Ballard's general Hough transform approach [70] since we do not take into account edge orientations. As such it relates more to Merlin and Farber's work [73]. We also think this is a more generic approach, since by not taking into account the edge orientation, the technique can be used for 3-D tracking. The algorithm can be extended to take into account different cues related to the edge features, like edge orientation or colour [74], and even different types of features simultaneously.

2.4.3 The randomised Hough transform Our review of the randomised Hough transform is begun with a reference to Fung *et al.* [75] that clearly illustrates the main ideas of the randomised Hough transform. The image is first filtered to obtain edge features. The edge orientation is used like in the generalised Hough transform presented by Ballard [70].

One of the major differences consists in using a many-to-one mapping: two points are randomly selected and, using a reference table indexed on the direction of the edge underlying the point, a previously selected anchor point of the object can be determined. The operation is iterated until the level of confidence on the solution is above a fixed threshold. Compared with the traditional Hough transform, that votes for many transformations compatible with a point, this technique is much faster since only one transformation is voted for in the transformation space. The transformation space is often referred to as the *parameter space* or, in relation with its implementation, an *accumulator space*. Usually a multi-array is used to store the votes that each transformation obtains which leads to a problem of storage when the number of dimensions of the transformation space increases. But since fewer transformations, or subsets of transformations to be precise, obtain votes when using the randomised Hough transform, another type of data structure can be used. For instance a hash-table can drastically improve storage requirements.

The transformation space that is being considered consists of translations combined with scale on the x-axis and y-axis, and thus has 4 dimensions. An extension is proposed to take into account rotations. However, it does not seem that shear transformations can be accounted for by using this technique since angles are modified by these transformations.

Another issue is that lines cannot be located since the technique uses a reference table indexed on the orientation of the edge point which is the same for all points of a line.

We mentioned that a many-to-one mapping has been used. The article does not specify what was done when, in the reference table, one orientation correspond to a few points. One possible solution is to calculate the corresponding anchor point for any points that have the same orientation, and it would thus be a many-to-few mapping. Alternatively it could be implemented by selecting randomly one point that has the same orientation. Except for this imprecision, the description in pseudo programming language of the algorithm is particularly clear and straightforward.

To sum-up, the key characteristics of the randomised Hough transform are: (1) only part of the image point features are used; they are selected randomly; (2) a many-to-one mapping is performed and (3) a list structure is used for the transformation space instead of an accumulator array.

Kälviäinen's thesis [76] provides more details and discussions on the randomised Hough transform (RHT). The literature review is of special interest. It completes the survey done by Leavers [67]. It is argued that the selection of an optimal and efficient resolution of the accumulator space are issues of the Hough transform (HT) and it presents several new extensions that have been proposed to overcome these issues. The literature review divides the presented techniques into two categories, non-probabilistic Hough transform and probabilistic Hough transform (PHT).

Gerig [77][78] presents a technique called back-transform in the first paper and back-mapping in the second. It consists of first performing a classical Hough transform and storing results in a first accumulator A. As for the standard Hough transform (*SHT*), each feature point is associated to a hyper-surface in the transformation space and the cells of the accumulator A that are intersected by the hyper-surface receive a vote. Once this has been done for all feature points, a second accumulator B is used, the hyper-surface associated with a feature point is evaluated again. Using A which stores the vote of the previous stage, only the cell(s) that intersect(s) this hyper-surface and having the maximum number of vote is/are considered. The corresponding cell(s) in B get(s) a vote and a reference to the feature point. Thus B becomes a one to many mapping of the transformation space to the image space. It is claimed in this paper that the number of false maxima is reduced and the interpretation of the accumulator space is considerably simplified. In addition, the feedback from cells in accumulator space to contributing feature points offers the possibility to apply more complex strategies to better identify shapes. Although the method may evoke the stencil approach that is presented in this thesis, the technique is different. The stencil approach consists in pre-evaluating the mapping of a shape given a reduced bounded transformation space such that an inverse map of the transformation space to the image space is obtained.

Li *et al.* [79] also refer to back-mapping but uses a different definition: the association to a hypercube of the transformation space to the image points that have triggered a vote for this hypercube. They present a hierarchical implementation of the Hough transform to detect lines in images or planes in range images. The implementation uses the k-d tree structure and the complexity of the fast Hough transform (*FHT*) algorithm is discussed in details. However efficient the implementation is, the application domain of the technique is limited since it requires the hyper-surface corresponding to a feature point in the transformation space to be hyper-planes. Consequently, the article concludes by highlighting the need to extend the *FHT* to higher order surfaces and/or to consider approaches that allow non-planar, possibly higher dimensional, problems to be recast as planar problems. The stencil approach described in this thesis allows a hierarchical approach of the Hough transform to be implemented for any templates we are looking for, parametrised or not.

Kiryati *et al.* [80] presented a theoretical and experimental comparison of the randomised and probabilistic Hough transform to detect lines. The *PHT* works like the *SHT* developed by Duda and Hart in 1972 [81]. Thus, it is a 1 to n mapping of the image space to the transformation space where only part of the points of the image space are considered. It is shown that, when the algorithms are used to detect lines, the *RHT* is better suited for high quality low noise edge images since it is considerably faster than the *PHT*. However, for the analysis of noisy low quality images the *PHT* is significantly more robust and the *RHT* can no longer be used. In terms of speed the *PHT* performs significantly better than the *SHT*.

2.5 Summary

Different approaches to track and identify object states have been presented. In this work, a model consisting of feature points of an object has been chosen. The object position is therefore characterised by the relative position of the features belonging to the object. This last point is further developed in chapter 4.

After initialisation, that can be helped by motion detection and/or background subtraction the object can be tracked, using a particle filter for instance as developed in chapter 3, to limit the region of interest where to look for the object. If distortions are not significant or corrected the set of feature points can be identified with the model using various matching techniques.

Variations of the Hough transform like the bounded Hough transform, that is called here the stencil estimator and that is presented and developed in chapter 5, allows to match, in real-time when the number of degrees of freedom of the motion can be limited, two sets of points in a bounded region that can be situated around the previous object state.

Chapter 3

Particle filters: a class of statistical methods for real-time tracking

3.1 Introduction

Particle filters are one of the main track that was followed to solve the real-time pose estimation problem that was faced. This chapter expands on the abstract review of particle filters as previously presented in section 2.3.2.A slightly different interpretation is proposed and discussed with a concrete example. In spite of being incomplete, since no Bayesian framework is taken into account, this interpretation does provide an intuitive explanation on how particle filters work, upon which a more precise understanding can be built.

Particles can be viewed as hypothetical states of the object being tracked. The validity of these hypotheses are quantified by measuring the image (say using a correlation measure, an edge-based matching measure *etc.*). These measures are then integrated as the weight of the particles and represent the validity of the hypotheses. The higher the weight, the more likely the hypothesis describes the state of the tracked object or at least a state close to the object position. This assumes that the measure has a certain degree of continuity which is preferable to obtain better results and which could allow, for the best hypotheses, gradient descent optimisation to be used to locate the object. This is a common improvement in particle filters that is sometimes referred to as smart particle filters [82].

To predict the hypotheses for the next frame, hypotheses are selected according to their likelihood, *i.e.* an hypothesis that is very likely to characterise the object state is selected multiple times while an unlikely hypothesis might not be selected, and are propagated according to the system model. The system model could consist of simply shifting the hypothesis by an amount proportional to the previous speed of the object, or by a more acurate albeit more complex model if the kinematics of the objects can be determined more precisely.

In the micro-pipette tip tracking scenario described later, the two features of a particle are the x and y location of the tracked object. The validity of each hypothesis was first evaluated by correlating a template image of the pipette tip with the part of the image centered on the hypothesised location. However, in the tracking of the pipette, pure correlation of the grey scale template image gave very poor results. The results were improved by filtering the image with an edge filter and correlating with an edge template. Figure 3.1 is a simplified flow chart that sums up the steps involved in the particle filtering algorithm.

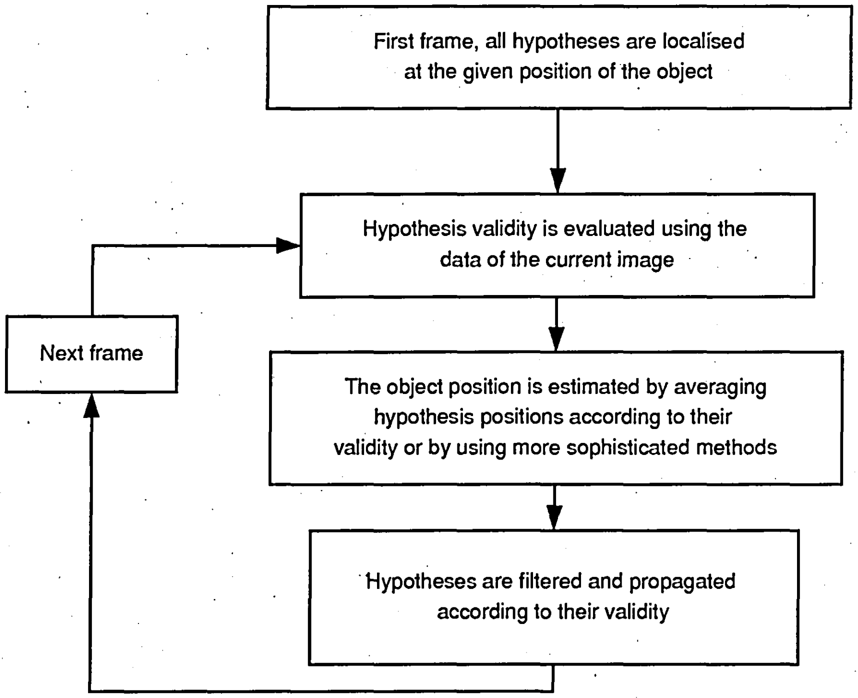
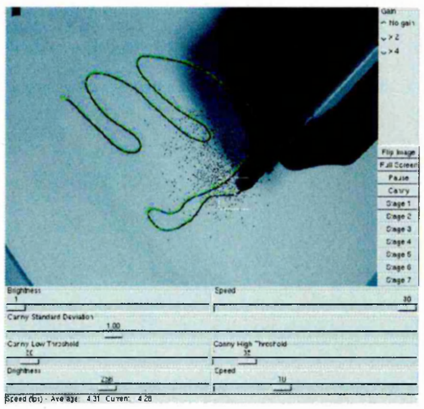

First frame, all hypotheses are localised at the given position of the object
Hypothesis validity is evaluated using the data of the current image
The object position is estimated by averaging hypothesis positions according to their validity or by using more sophisticated methods
Hypotheses are filtered and propagated according to their validity
Next frame

Figure 3.1: Particle filtering stages.

3.2 A real world example

In this study an existing implementation of particle filters has been used [1]. The measure used is the correlation between a template image that represents the object to track, here a pen tip, and the part of the image around a particle. Here, particles are 2-D points indicating the position of the image part that has to be correlated. The estimation of the pen tip position for a given image is evaluated by averaging the particles positions weighted by the correlation measure. Figure 3.2 shows the program at the end of its execution. Green dots represents the tracked location in previous images. They are located on the trajectory of the pen tip that appears due to the line that was drawn by the pen. Black dots in the surrounding of the pen tip represent the particles positions for the last frame. It can be seen that they are not spread uniformly around the pen tip. This is due to the hand shadow that correlates well with the image template. Nevertheless, the pen tip is successfully tracked.



Brightness 1
Speed 30
Canny Standard Deviation 1.00
Canny Low Threshold 20
Canny High Threshold 35
Brightness 256
Speed 10
Speed (fps) - Average: 4.31 Current: 4.28

Figure 3.2: The original pen tip tracking algorithm using particle filters [1].

3.3 Typical encountered issues

The above mentioned implementation was adapted to track a micro-pipette tip for biological cell manipulation. The tracking of the pipette tip failed quickly as shown in figure 3.3. The image sequence has been captuered using an optical microscope. The white objects are cell holders that use electo-static charges to trap cells at the circular locations.

Before explaining the reason for the failure of the tracking a few images of the pipette sequence are presented. A short historical account of the investigations is then reported. Figure 3.4 shows a few images from the pipette sequence that illustrates common issues related with tracking. Frame A is the first frame of the sequence where the template image of the pipette tip is captured. Frame B demon-

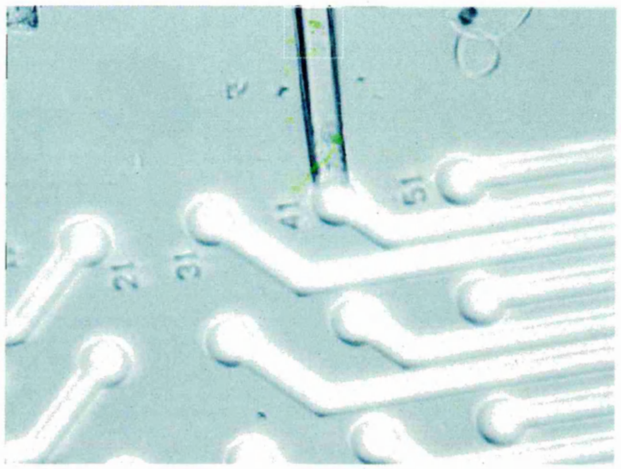


Figure 3.3: The tracking of the pipette tip has failed. The white square is the tracked location (at the top centre position of the image). Top left hand corner is the template image.

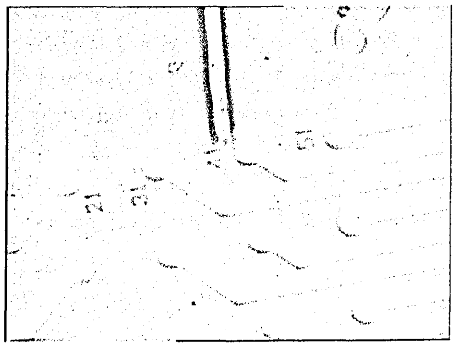
strates the first problem encountered: motion blur. In frame C, a problem quite specific to our video sequence appears: the pipette tip is transparent and its appearance changes with the background which likely results in poor similarity measures with the initial template image. In frame D the illumination of the scene has changed *i.e.* the scene progressively gets darker. Finally, in frame E the whole scene is blurred.

3.4 Preliminary approaches to solve these issues

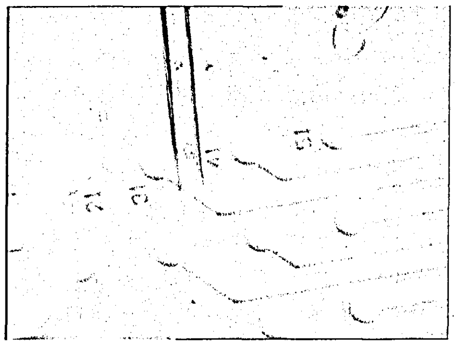
The different approaches that have been considered in solving the problems above are as follows:

- updating the template image. This led to the following problems
  - the periodicity of the update needs to be decided.
  - when an update is performed parts of the background is incorporated into the template. Also, due to the transparency of the pipette, the backgroung can alter the look of the pipette.
- accounting for the orientation of the pipette.
- enhancing the particle filter.
- improving the matching method as well as integrating new matching methods.

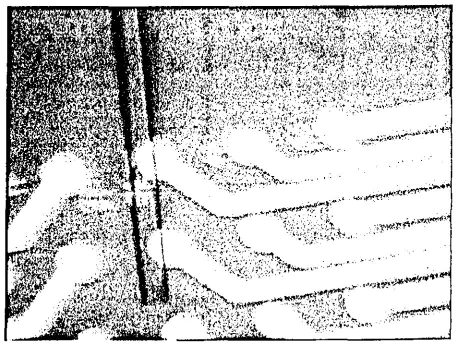
Frame A: the initial position



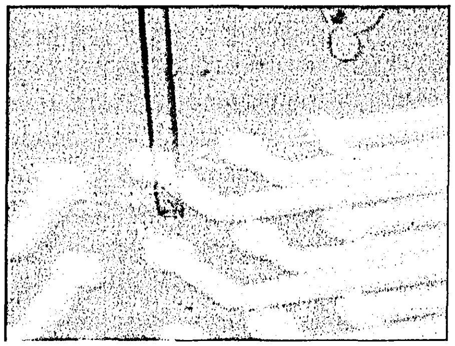
Frame B: motion blurred micro-pipette tip



Frame C: the changing background behind the micro-pipette tip modifies its appearance



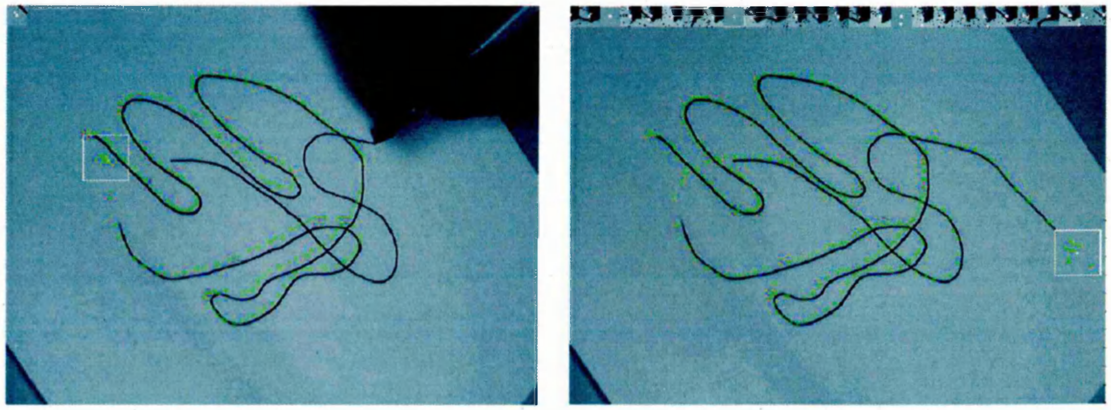
Frame D: the illumination level of the scene has changed



Frame E: the whole scene is blurred due to defocussing

Figure 3.4: Problematic frames from the micro-pipette sequence.

In hindsight, the problems were not correctly identified, and so relatively complex methods were initially developed. This is a trap that is easy to fall into, especially in computer vision. This was due to the fact that the issue was not directly apparent, like the change of illumination of the scene was. We will proceed with the account of the steps we undertook to try to solve this issue.



Frame A: Updating the template image without storing the previous template images.

Frame B: Pen tip tracking with update of the template images. The previous template images are displayed at the top of the image.

Figure 3.5: Tracking with the template updating mechanism running

3.4.1 Dynamic template updating The tracker was improved by firstly implementing a template update scheme. It was tested on the pen tip sequence as shown in figure 3.5. Nevertheless, when applied to the pipette sequence, tracking continued to fail after a few frames. The major issue is that we do not have any criteria to decide when the update has to be performed. Thus there is a possibility for the update to be performed on a part of the image where the object is not present. It was later realised that by combining different measuring methods, sensitive to different variation of the pipette tip appearance, it may be possible to update the template. If measures start to give low values while others continue to give a high confidence value, it indicates that it is time to update the model associated with the measure giving low values. However, since this continued to prove inconsistent, the idea was discontinued.

3.4.2 Taking measures A few other improvements were experimented with until it was decided to systematically take measures of the image sequence in order to better understand why the tracking was failing. Thus, the correlation of the template image with the scene image, over a predefined region of interest, was measured. In other words, the template image was convolved with an image region.

Figure 3.4.2 shows the scene images on which the correlation measure has been displayed with grey shades. In order to improve the visibility, we used the following colour code: black was used to mark the points when the correlation measure is higher than 0.95 times the value of the best correlation measure in the region surrounded by the large rectangle. White was used to mark the points with a correlation measure in the range 0.9, 0.95 times the the best correlation value. Next ranges were coloured in grey then white again and then transparent colour was used to see the underlying image. Therefore, a small set of black points surrounded by white indicates a good candidate location for the pipette tip. We focused on a few images where the tracking of the pipette was systematically failing. From the pseudo-coloured frames of figure 3.4.2 the following observations can be made.

Frame A shows that the cell holder on the background is highly correlated with the micro-pipette tip template. The left boundary of the micro-pipette has a good correlation value relative to the rest of the region. Note that the micro-pipette tip is not yet present in the large rectangular region. However within this region, the top of the cell holder correlates quite well with the micro-pipette tip. This is confirmed by frame B.

In frame B it can also be seen that even though the tip of the micro-pipette is correctly located, the correlation values along the body of the micro-pipette are high as well.

Frame C and D again illustrates the fact that the body of the micro-pipette has got a high correlation value and in Frame D the micro-pipette failed to be detected because of this phenomenon. It appears that according to the correlation measure the micro-pipette tip could be located at three different positions. Although the particle filter copes well with multimodality, the evaluation of the micro-pipette location, that is made by averaging the location of the particles with their weight, lead to an incorrect result.

Frames E and F present two other examples where the tracking is lost. Because of the background, the best correlation is no longer found on the micro-pipette

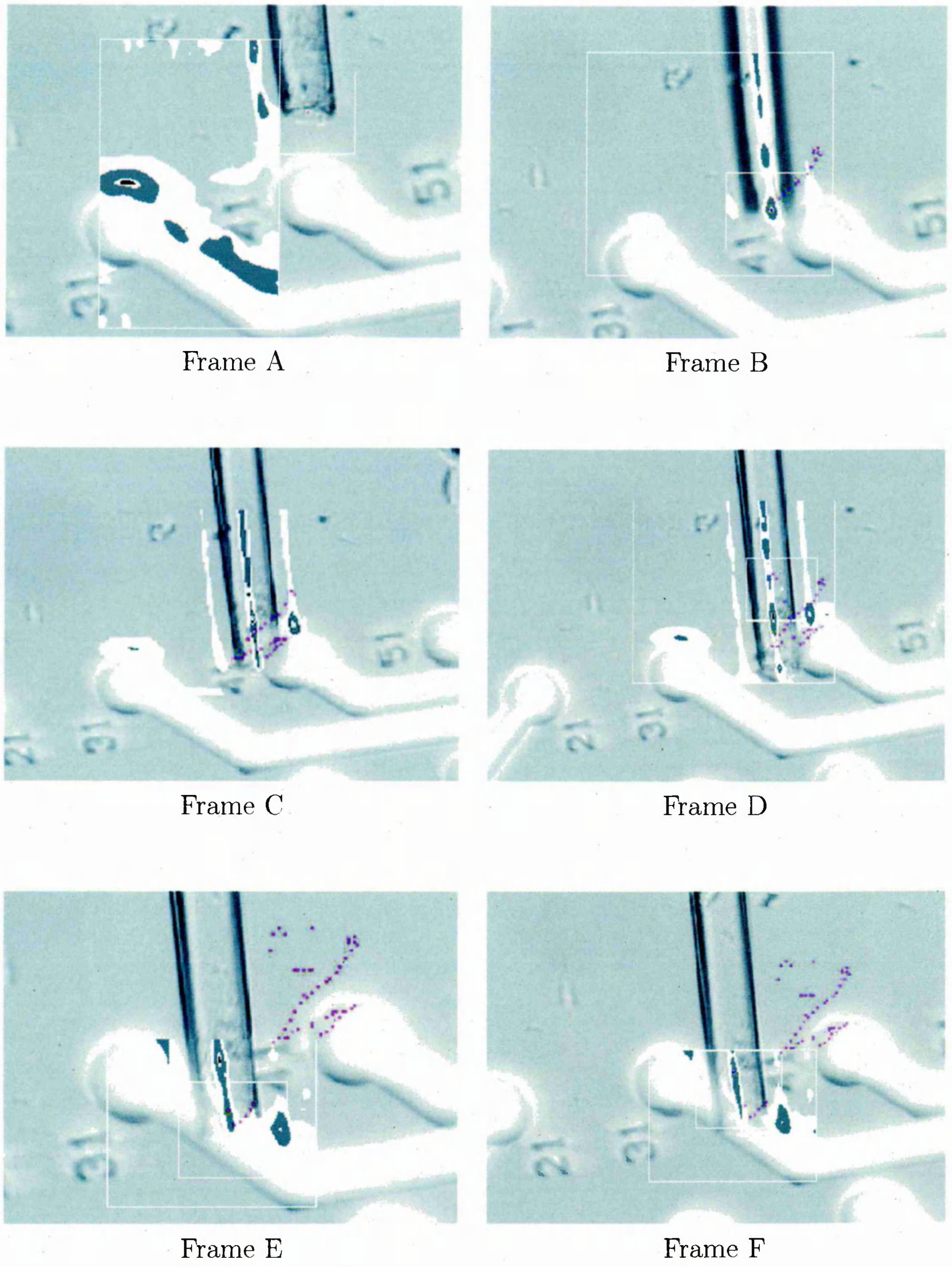


Figure 3.6: Image showing the value of the correlation measure within a region of interest. The positions with high correlation values are coloured following the colour code described in page 44.

tip. Although the particle filter can cope with occlusions the particle filter cannot recover because the micro-pipette tip is occluded for too many frames and when the micro-pipette tip appears again, in a region further down, because clutter prevented particles spreading, no particles were present in that region that could have made the recovery possible.

3.4.3 Improving the matching paradigm Once the problem was clearly identified as being the presence of clutter, a simple solution to deal with it was quickly found. To improve the quality of the tracker a measure that better discriminates the micro-pipette from the background clutter was required. We found that correlating edges, instead of grey scale images, greatly improves the matching. Indeed, edges of the pipette is a good characteristic to identify the micro-pipette since the body of the pipette changes due to its transparency. Frames A,B,C,D of figure 3.7 illustrate that the tracking of edge filtered images has been greatly improved.

Comparison of some edge detectors.

When it was observed that edge correlation was superior than the grey level template matching, different edge detection algorithms were evaluated.

We tested a variety of edge detector methods including the Laplacian of a Gaussian LoG[83], Canny[84] and Susan[85]. Canny with the appropriate parameters is more consistent and faster than LoG and because of its thinning edge step, edges are better located and thinner than LoG. The Susan edge detector method is even faster than Canny (by a factor of 10) and allows the particle filter to better locate the micro-pipette. Although software patents are not currently legal in Europe, Susan's major drawback is that it is a patented technique. Figure 3.4.3 illustrates results obtained with the Susan and Canny edge detector.

Optimisation and pitfalls

Edge template matching using correlation, because it involves an additional step, is slower than direct grey-level correlation. Since real-time performance is a prerequisite for us we explored the possibilities of optimising the edge detection process. Therefore the whole image was first filtered. To improve even more this scheme the egdes were extracted where measures were taken. Figure 3.4.3 shows the results

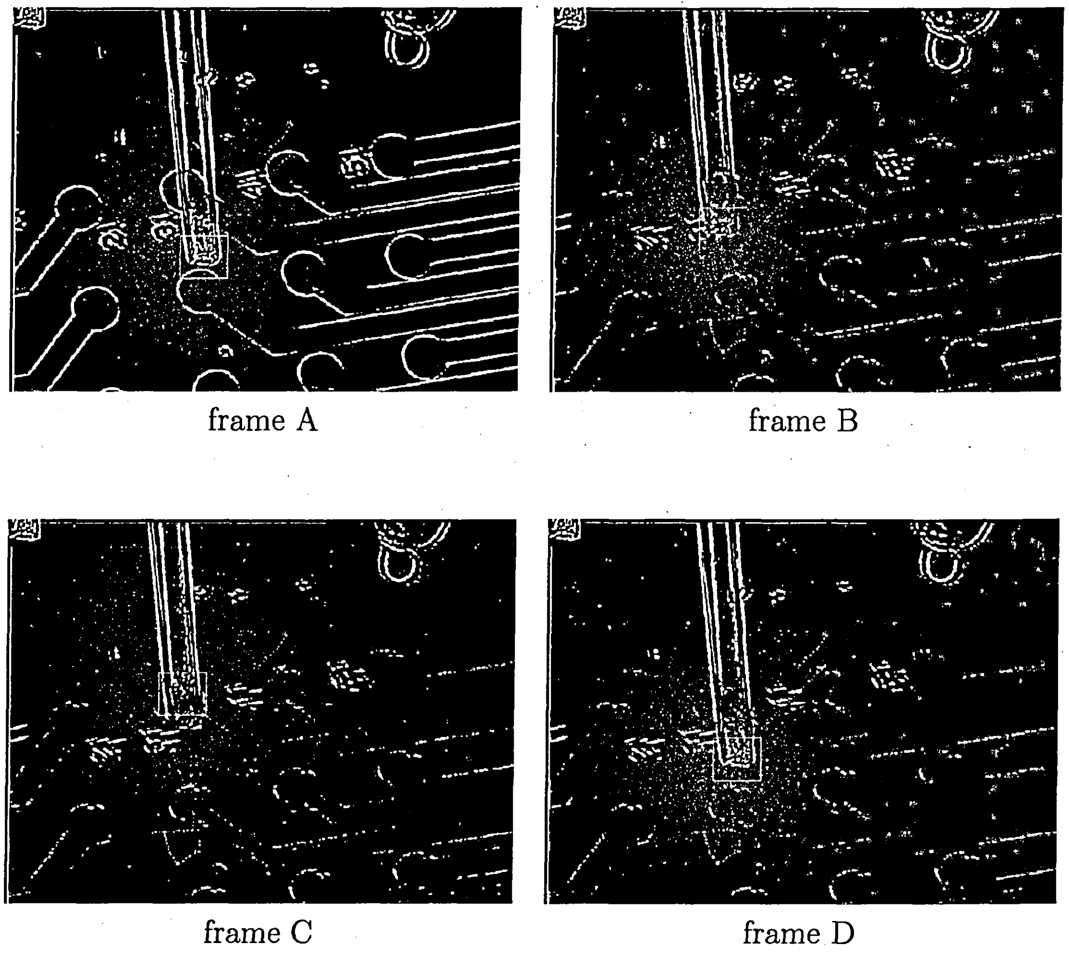
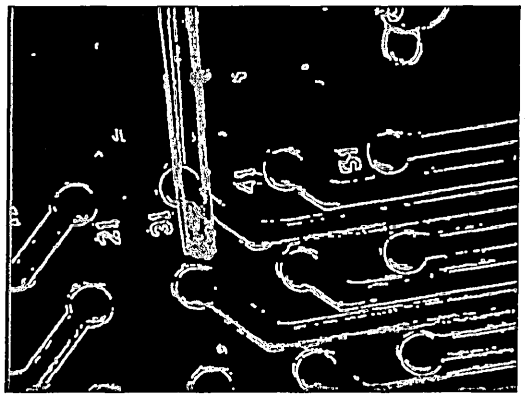


Figure 3.7: Frames showing the tracking results using edge template matching. The top left corner of images shows the edge template of the micro-pipette tip.


frame A: one image filtered with the Susan edge detector.

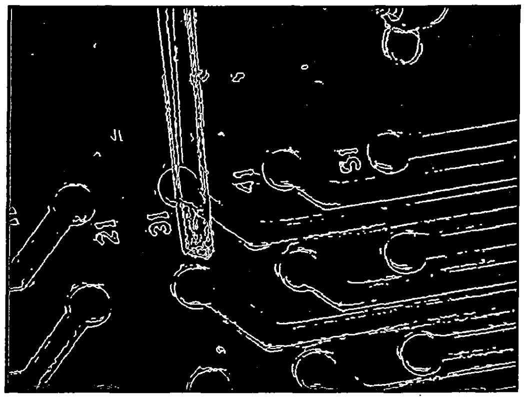

frame B: the same image filtered with the Canny edge detector.

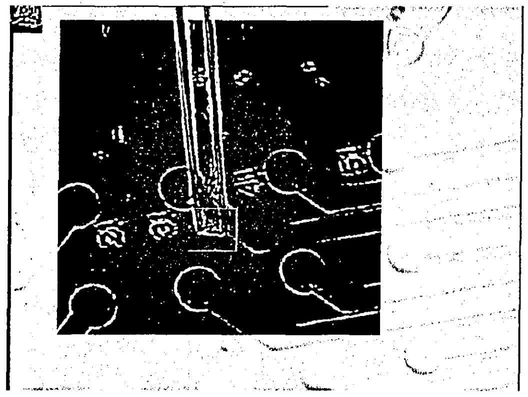
Figure 3.8: The same image filtered with the Susan and the Canny edge detector.

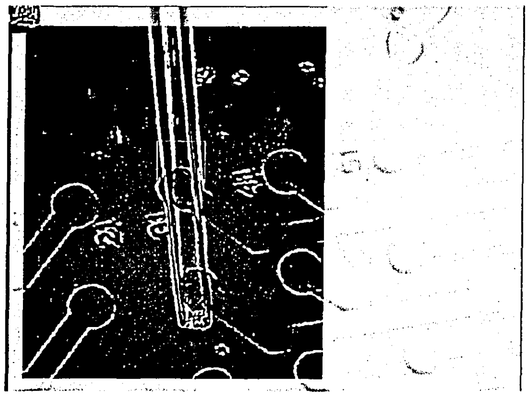
of this operation. The speed improvement due to this last optimisation is not significant because the reduction of the filtered area is not huge. This illustrate the classical mistake of attempting to optimise too much, too early on without profiling a program.

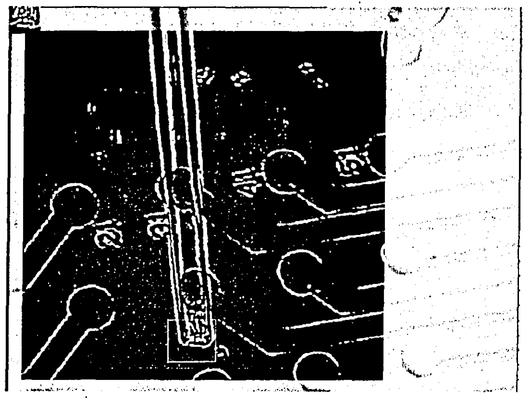
3.4.4 Geometric Branch-and-Bound Matching The edge correlation method to match the model template with the image was still not completely satisfactory: the tracking was still failing towards the end of the video sequence and the recovering time, when the track was lost, appeared to be too long. To improve the tracking we decided to further improve the matching measure. The geometric branch-and-bound matching method was selected for that purpose. For a description of the branch and bound (BB) algorithm please refer to the literature review on page 19.

Implementation

A Java implementation of the BB algorithm was provided by Breuel [35] but for compatibility reasons it was re-implemented in C++. The original Java code takes into account the change of orientation of the object as well. But our implementation excludes this feature.


frame A: the particles are widely spread on the image.


frame B: the tracking fails.


frame C: but is able to recover thanks to the particle located near to the tip.

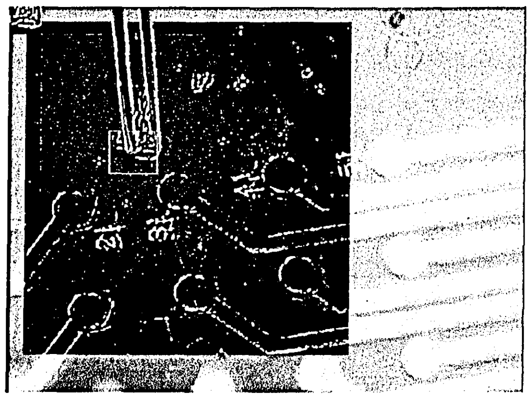

frame D: end of the tracking sequence

Figure 3.9: Frames showing the results of edge correlation with only part of the scene image processed in order to filter edges. Frame B illustrates one of the frames where edge correlation fails.

Testing

One of the major issues of the BB algorithm is its speed. Determining theoretically the average and worst case complexity of these kinds of matching algorithms remains an open problem[35]. However, BB was found to be more time consuming than performing an exhaustive search on an area of the same size using a correlation measure. This supposedly depends on the area size selected. Nevertheless, adding other cues such as edge orientation, or colour can speed up BB.

In order to incorporate BB with the particle filter, a measure to quantify the goodness of match of the template image and the scene image had to be provided. The ratio defined by the number of edge points matching the image by the total edge points of the template image was used.

The usage of BB led to good tracking results but the tracking was too slow for real-time applications. Enhancements by incorporating the orientation and scale of the template can be taken into account by the BB algorithm. Also, the BB algorithm can be extended to take into account different kinds of features simultaneously, *e.g.* colour points along with corner and edge features. BB is thus a good candidate if different measures need to be combined and it was used as a complementary measure as explained further down.

3.5 Clustering particles

3.5.1 Locating the statistical modes In particle filtering based methods, the predicted location of the object being tracked can be obtained by using a weighted mean of the particles' positions (equation 2.3). When working in a cluttered environment, the *pdf* is bound to be multi-modal. This is because similar objects in the background gives a probability measure which is very similar to the probability measure at the actual location of the object. As a consequence the weighted mean of the particles position is not relevant anymore to identify the position of the tracked object (figure 6.44 and 6.45). What needs to be done is to isolate the different modes of the *pdf*. Once the modes are isolated, they will provide a much reduced set of relevant probable locations of the object. On these probable locations, more accurate, albeit more time consuming, measures can be applied to decide which of the modes is the actual location of the object. To isolate the different modes we defined the following methodology.

3.5.2 Method formalisation

Let

- $S = \{s^{(1)}, \ldots, s^{(n)}\}$ be the set of n particles obtained using particle filtering. $s \in \mathbb{R}^n$ the state space.
- $T = \{s \in S | \text{ev}(S) > \textit{threshold\_value}\}$ the subset of most relevant particles (samples, hypotheses). $\text{ev} : \mathbb{R}^n \mapsto \mathbb{R}$ is the function that evaluates the validity of the sample. $\text{ev}(s^{(i)}) = \hat{\pi}^{(i)}$, where $\hat{\pi}^{(i)}$ was defined in page 28.
- $t \in T$ one of the thresholded particles
- $M^{(i)}$ a subset of T
- $\|.\|$ the Euclidean distance in $\mathbb{R}^n$

T is partitioned as follows

$$\forall t \in T : \exists! i \in \mathbb{N} : t \in M^{(i)} \tag{3.1}$$

$$\forall (t^{(j)}, t^{(k)}) \in T^2 : \left(\|t^{(j)} - t^{(k)}\| \leq d \Rightarrow \exists! i \in \mathbb{N} : \{t^{(j)}, t^{(k)}\} \subseteq M^{(i)} \right) \tag{3.2}$$

The partition of T depends on the distance d and the threshold value. Since multiple partitions satisfy these two conditions we consider the one that has the maximum number of elements.

$M^{(i)}$ are sets of relevant hypotheses belonging to the same neighbourhood. As such, they are characteristic of the presence of the tracked object, we call them *mode* sets. To optimise the use of the measurements the non-thresholded samples h are integrated within a mode set $M^{(i)}$ *iff* $\min_{s \in M^{(i)}}(\|s - t\| \leq d)$. Note that samples bordering mode sets might belong to more than one set and that some particles are discarded from the process.

Ultimately, the weighted average of the elements belonging to a mode is evaluated

$$\forall M^{(i)}, m^{(i)} = \frac{\sum_{s \in M^{(i)}} s.\text{ev}(s)}{\sum_{s \in M^{(i)}} \text{ev}(s)} = \sum_{s^{(j)} \in M^{(i)}} s^{(j)}.\pi^{(j)} \tag{3.3}$$

A graphical representation of the process is shown in figure 3.10.

3.5.3 An $O(n)$ algorithm to cluster particles

A practical algorithm, with linear complexity, to cluster samples is now introduced. The underlying idea of the

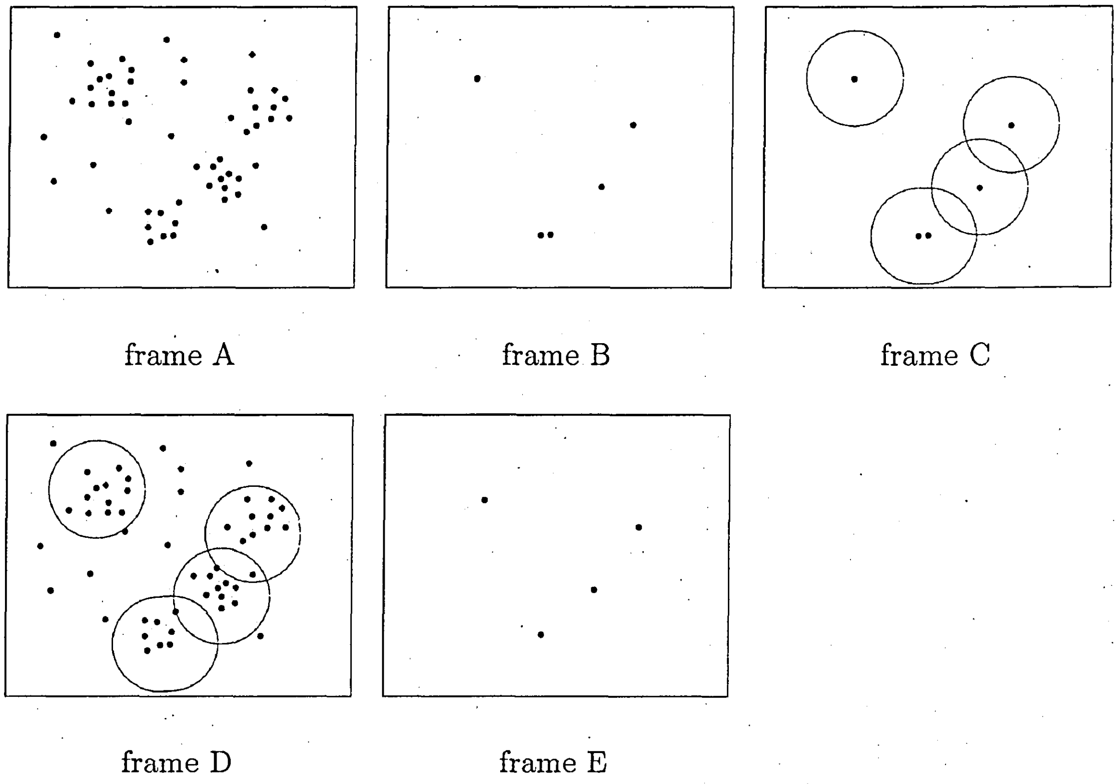


Figure 3.10: In frame A points represents particles. Frame B shows the thresholded particles. In frame C, the boundaries show how these particles are grouped and can be seen as the set bounds . Frame D shows how the other particles are collated into the corresponding sets. Frame E shows the resulting peak points, compared with frame B the modes are shifted from the thresholded particles and the 2 closed thresholded particles have been integrated in the same mode.

following heuristic method is to collate particles, starting with one located in the neighbourhood of a mode and then to recursively cluster particles located near to it.
Let:

- a mode be a particle associated with a weight and as such an element of $\mathbb{R}^n \times \mathbb{R}$.
- the weighted mean function of two hypotheses associated with their validity measure be

$$\begin{array}{rccc} \mathrm{wm}: & (\mathbb{R}^n \times \mathbb{R}) \times (\mathbb{R}^n \times \mathbb{R}) & \mapsto & \mathbb{R}^n \times \mathbb{R} \\ & ((h_1, w_1), (h_2, w_2)) & \rightarrow & (\frac{w_1 h_1 + w_2 h_2}{w_1 + w_2}, w_1 + w_2) \end{array} \tag{3.4}$$

- $\|.\|_a$ be the Euclidean distance between a particle and the n first elements of a mode.

We define recursively the set M of modes on the set of samples S as follows:
$M := M_{card(S)}$
$M_0 := \emptyset$

$$\begin{array}{rll} M_k := & \{\mathrm{wm}((s^{(k)}, \hat{\pi}^{(k)}), m) \mid \|(s^{(k)}, m)\|_a \leq d, m \in M_{k-1}\} \\ \cup & \{m \in M_{k-1} \mid \|(s^{(k)}, m)\|_a > d\} \\ \cup & \{(s^{(k)}, \hat{\pi}^{(k)})) \mid \\ & (\forall m \in M_{k-1},\ \|(s^{(k)}, m)\|_a > d)_\wedge (\hat{\pi}^{(k)} \geq \mathit{threshold\_value})\} \end{array} \tag{3.5}$$

At step k of the construction of M, the k^{th} sample is considered. It is merged with an existing mode as described by the first set of formula 3.5. Otherwise, if this is not possible, due to the distance constraint, and if its measure is high, it is considered as a new mode as described by the third set in formula 3.5. The second set in formula 3.5 retains unchanged modes for the next recursion stage.

The resulting modes depend on the particle selection order and thus are different from the modes $m^{(i)}$ as previously defined. An improvement can be made by ordering particles according to their reliability, so that hypotheses located near to an undiscovered mode are not discarded and that the drift of a mode due to the subsequent incorporation of particles is minimised. The best complexity for a sorting algorithm is $O(n \log(n))$ which in practice is not an issue since n is small for particle filters. Sorting particles provides deterministic results for the resulting modes, however, in practice, it remains to be shown that this is an advantage.

Figure 3.5.3 shows the test of an implementation of the above algorithm. In frame A, there are some modes in the body of the micro-pipette. This indicates that the body of the micro-pipette is very similar to its tip. Frame B shows the tracking just before the program looses the micro-pipette tip. Frame C, shows the tracking of the micro-pipette having recovered after a long period of time where the tracking was indicating the pipette body instead of the tip. Again, the modes in the pipette body indicate that the body is very similar to the pipette tip. The tracking of the micro-pipette eventually fails as shown in frame D.

3.5.4 Location of the micro-pipette using further measurements of the image Summing up the processing steps thus far:

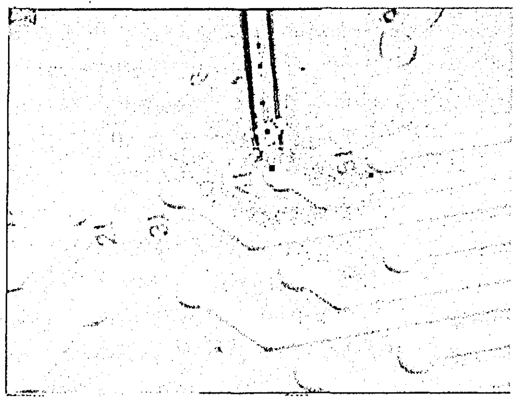
- The scene image is filtered using an edge filter.
- Particles giving the probable locations of the object are generated using the particle filter. A correlation measure of the edge template with the edge filtered image is used to weight particles' importance.
- The modes of the underlying *pdf* are located by collating, the weighted particles into groups.

At this stage two scenarios are possible:

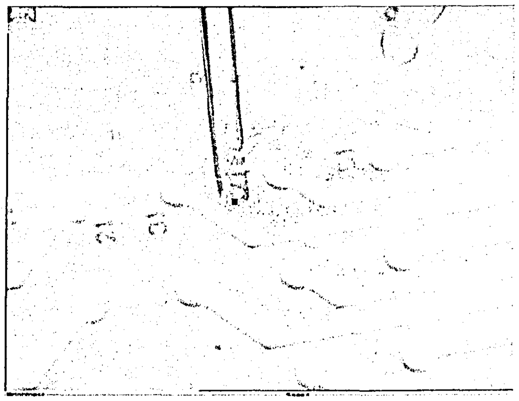
- If the object is not occluded and if the scene image is not cluttered, it is likely that the mode having the largest weight indicates a position close to the object position.
- However, if the object is being tracked in a noisy environment, it is probable that multiple modes with very similar weight are found.

In the second case, illustrated by figures 6.44 and 6.45, further processing is needed to decide which of the modes gives the most probable location of the object.

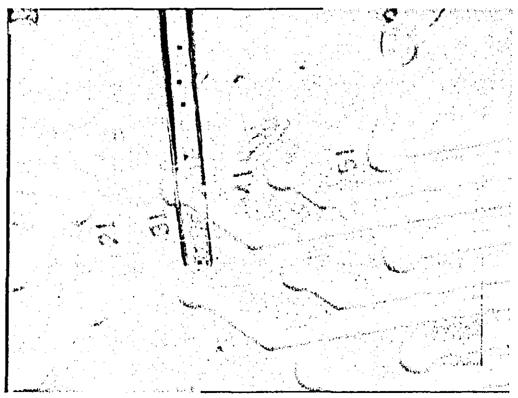
For that purpose, the branch-and-bound algorithm was performed in a region surrounding the detected modes. BB was performed using edge filtered images, the ratio of matched points over the total number of edge point of the template was the associated measure. To further discriminate the modes the surrounding region was



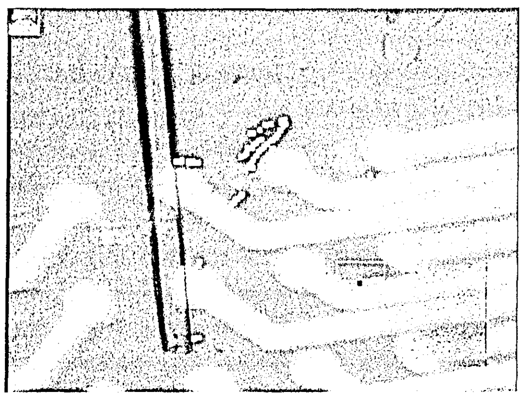
Frame A: Large black squares are the modes.



Frame B: Large white dots are the tracked locations.



Frame C: A large number of particles are very far from the micro-pipette tip.



Frame D: Tracking is lost, particles are on the bottom right corner.

Figure 3.11: Tracking of the micro-pipette tip. Modes give an indication of the location of the peaks of the *pdf* sampled by the particles.

correlated with a grey level template image of the object. A combination of the two measures was used to select the appropriate mode: we attributed a weight to each of the measure and took the barycentre of the measures.

3.6 Further improvements

3.6.1 Overcoming the effect of clutter Reducing the spatial range of particles helps to avoid the effect of background clutter. When particles are bunched around the object position, the evaluation of the *pdf* sampled around the likely object position has a better chance to be unimodal (figure 6.43). The following method has to be performed cautiously in order to maintain the robustness of the particle filter. The improvements discussed cannot always be applied but when conditions are favourable they improve the reliability of the filter.

3.6.2 Integrating the kinematics of the object Based on the previous tracked location of the object its current speed and direction can be evaluated. An additional weight is given to particles describing object positions compatible with previous estimated kinematics. The additional weight augments the actual measure calculated from the image boosting up the weights of the particles compatible with the previous object kinematic and reducing the effect of the background clutter.

For the pipette problem, first attempts in integrating the velocity (speed and direction) failed because the pipette movement is too random. However, the speed of the pipette tip is more or less constant and integrating this data to the particle weights helped discard high value measures due to background clutter.

Integrating the speed of the pipette tip using this method has the additional advantage that more particles are better situated. This better localisation of particles can be explained by the fact that since particle weights around the predicted object location are augmented the corresponding particle are more likely to be selected and propagated to the next frame. Consequently, those particles which are further away from the predicted location are more likely to be left out. It avoids processing those locations where background clutter may play a disruptive role.

In order to avoid affecting the robustness of the particle filter these additional weights must be modulated by the degree of confidence that we have in the tracking. After a few frames, when tracking is lost, no additional weight should be used and in cases of uncertainty in the tracked location the additional weights should be reduced and given relatively to the last previous location that can be trusted.

3.6.3 Partial re-initialisation Partial re-initialisation of the particles is another proposed scheme that could be adopted to have a better sampling of the *pdf* of the object position. Partial re-initialisation assists the particle filter to better sample those regions of the state space that are highly likely to contain the tracked object position. The sampling of the remaining of the state space becomes sparser, but to a minor extent, ensuring that alternative possible states are still taken into account. Partial re-initialisation is particularly useful when the tracking fails due to occlusion. Indeed, when the tracking fails particles tend to spread across the state space which is the adequate behaviour to be able to recover the tracking of the object when these one reappears. However, when the object reappears because the number of particles is small around the object, it is possible that, even if a few particles are able to correctly locate the object, their number is not high enough to be propagated correctly to the next frame. In order to help the particle filter recover more quickly and surely, the propagation of the particles can be modified by relocating more particles to the supposed recovered location. This also reduces the likelihood of mislocation due to clutter in subsequent frames.

When particles indicate that the object location has been recovered, the partial re-initialisation is done as follows. Amongst the current set of particles, a percentage of particles having lower weights, and hence more likely further away from the current probable tracked location, are randomly chosen and are re-located to the current probable tracked location of the object.

These two methods, weight addition and partial re-initialisation, help to integrate the additional information brought by measures made on the modes of the *pdf*. The main issue is that they have to be fine tuned, the additional weight has to be determined as well as the percentage of particles to be relocated in order not to affect the robustness of the particle filter. Relatively small changes suffice to affect the particle filter behaviour since a better sampling of the *pdf* has the positive consequence of a better sampling of the *pdf* in the next frame and so on iteratively.

Figure 3.12 is a simplified flow chart that sums up the stages of the tracking procedure.

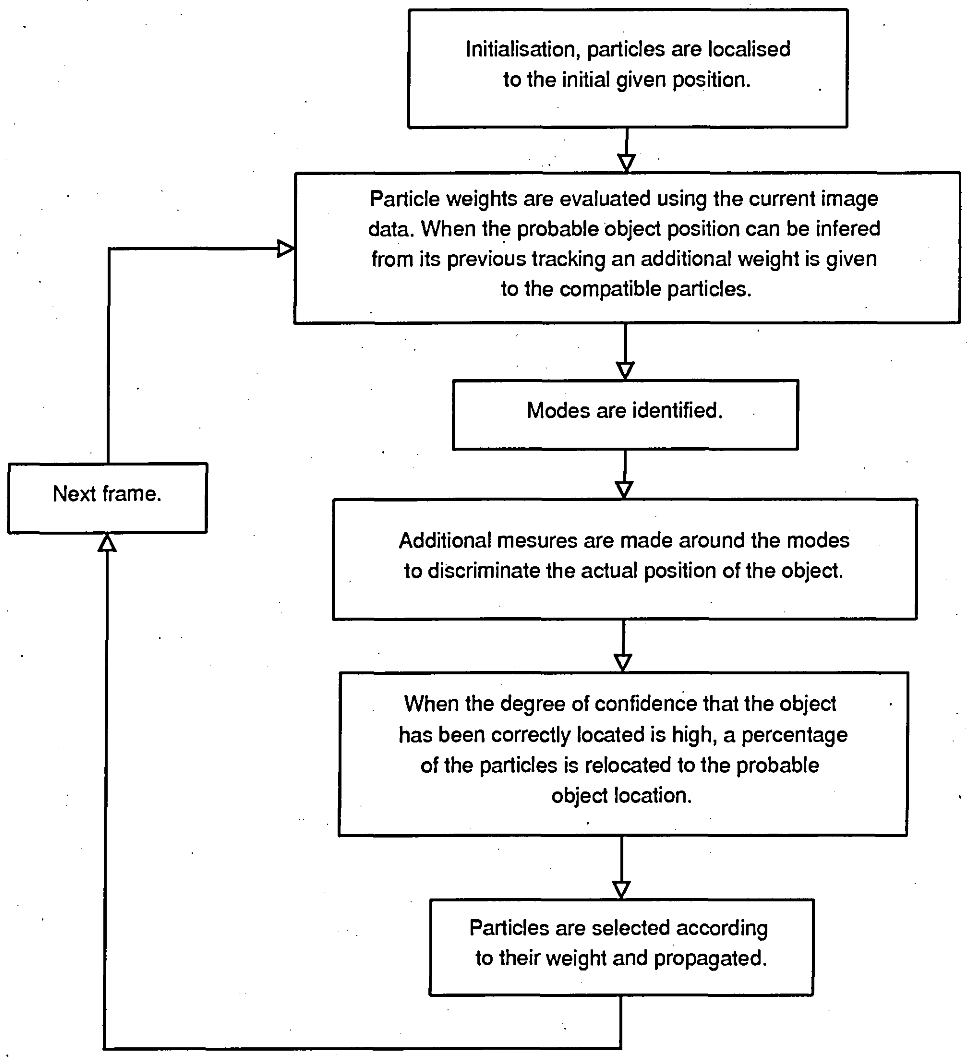

Initialisation, particles are localised to the initial given position.
Particle weights are evaluated using the current image data. When the probable object position can be infered from its previous tracking an additional weight is given to the compatible particles.
Modes are identified.
Additional mesures are made around the modes to discriminate the actual position of the object.
When the degree of confidence that the object has been correctly located is high, a percentage of the particles is relocated to the probable object location.
Particles are selected according to their weight and propagated.
Next frame.

Figure 3.12: Tracking stages.

3.6.4 Modes filtering Tests indicate that the method described above performed well. The tracked position sometimes jumped to a far off location from the micro-pipette tip. To avoid this, another scheme was added to the tracking algorithm as described by the figure 3.12. The idea was to consider only the modes close to the previous tracked location. This has a second advantage of avoiding the need to perform measurements around modes far off from the pipette tip, thus speeding up the application. Since the tracking of the pipette can fail, to be able to recover, far off modes of the previous location have to be taken into account. The following procedure has been used:

- only the modes at a distance d from the previous tracked location were considered.

- if the measures are high, indicating that the pipette tip is currently being correctly tracked, then the distance d is reset to its initial value.
- otherwise it is assumed that the tracking is lost and the distance d is increased.

This further improves the tracking of the pipette tip and illustrates an additional way to incorporate high level information such as the tracked location of the pipette, as opposed to its *pdf*, to a higher filter stage, *i.e.* the modes stage, as opposed to the particles stage.

3.7 A generic particle filter architecture

We re-implemented the CONDENSATION filter from scratch. The CONDENSATION algorithm is described in [54] and is a particular type of particle filter. Using the template mechanism provided by C++ we developed a generic implementation that can be adapted to track objects in different scenarios.

3.7.1 Description of the architecture The underlying design principle that has been retained was flexibility. Therefore no arbitrary choices were taken and whenever a choice has to be made the implementation takes it into account in order for users to make their choices without modification of the core architecture. Implementation wise these choices appear in functions, template parameters and policy idioms. The negative consequence is that it introduces a higher degree of complexity to the architecture. However, this degree of complexity can be made transparent by subsequent software layers.

3.7.2 Issues related to the initialisation of the class A brief description of the architecture choices and their rationale is provided here. The Doxygen[1]-commented implementation has been released in the Mimas C++ open source computer vision library. Examples of usage of the particle filter implementation, renamed hypothesis filter, are provided in the examples of the version 2 of the library.

An architecture based on the policy idiom has been adopted. One of our motivations, albeit controversial in justifying its choice, was to try to implement the policy idiom, a design pattern described in [86]. For more on design patterns one may refer

[1]A tool to document source code

to the famous, at least amongst the software designer community, "Gang-of-Four" book[87].

More reading on aspect oriented programming [88] and meta programming [89] [90] is useful to understand the ideas behind the policy idiom. Compared with the inheritance mechanism, because classes are assembled at compile time, more flexibility is provided; it is possible to define optional functions in the so-called host class that can be instantiated only if the policy classes implements a certain interface. For instance, in the `hypothesis_filter` class the `track()` method is instantiated only if the `image_loader` policy is providing a `next()` method and the `analyse_result` policy an `analyse()` method.

Nevertheless, although critical sections, such as correlation measuring algorithms and image filtering, still have to be written, due to performance issues, in a low-level language such as C++; it is preferable to use higher level languages having meta-language facilities, such as Python or Ruby, to implement generic logic for combining sub-systems. One may have a look at the required complexity of the code needed to implement meta programming in a language not conceived for this purposes, namely C++[89][90]. This being said, Koethe [91][92] provides matter to think about re-usability of code in C++. Koethe specifically discussed implementations relating to image datastructures for computer vision.

Adoption of a policy idiom based architecture brought to our attention the following issue: policies, such as the `observer` policy, can be quite different and need different initialisation parameters. However, the policy architecture requires the constructor prototype of policies to remain the same. In order, to solve this issue an embedded class, called `set`, is provided within each of the policies. For each of the policies this class has to be instantiated and given to the host constructor. This methodology has also the advantage of guaranteeing that policies are instantiated before the host class. Another advantage is that the order of call of the different policies constructors is done at the level of the host constructor, therefore the user does not have to worry about this. To our knowledge this method has not been previously documented.

3.7.3 Conception The tracking algorithm has been divided into five, mostly independent, components. The `picking` policy which ensures the propagation of the particles (hypotheses). The `observer` policy which has the responsibility for measuring the validity of an hypothesis (the weight of a particle). And which depends on the type of hypothesis selected, *e.g.* a 3-D location, a 2-D location associated with scale. The hypothesis type can be considered as a component, even if it not a policy. The `analyse_result` policy which is used to evaluate the position of the tracked object using the observation made on the image. The `compensation` policy that has been used to implement partial re-initialisation of the particle filter. `image_loader` policy which just facilitates the loading of images.

A previous implementation had fewer components that we called hypothesis filter, observer and custom hypothesis. The role of the particle filter was to propagate particles according to the system knowledge. The role of the observer is to update the knowledge of the system by taking measures on the image. The determination of the kind of hypothesis allows the filter to be adapted to the state space with which we want to operate: 2-D translation, 3-D moves, n-dimensional contours, *etc.* Figure 3.14 presents a UML diagram of the system.

The particle filter implementation was tested on the micro-pipette sequence, the pen sequence, 3 ping-pong sequences and a Rubik's cube sequence. These tests allowed us to refine the architecture and the implementation of the hypothesis filter, they are presented in section 6.3 page 143.

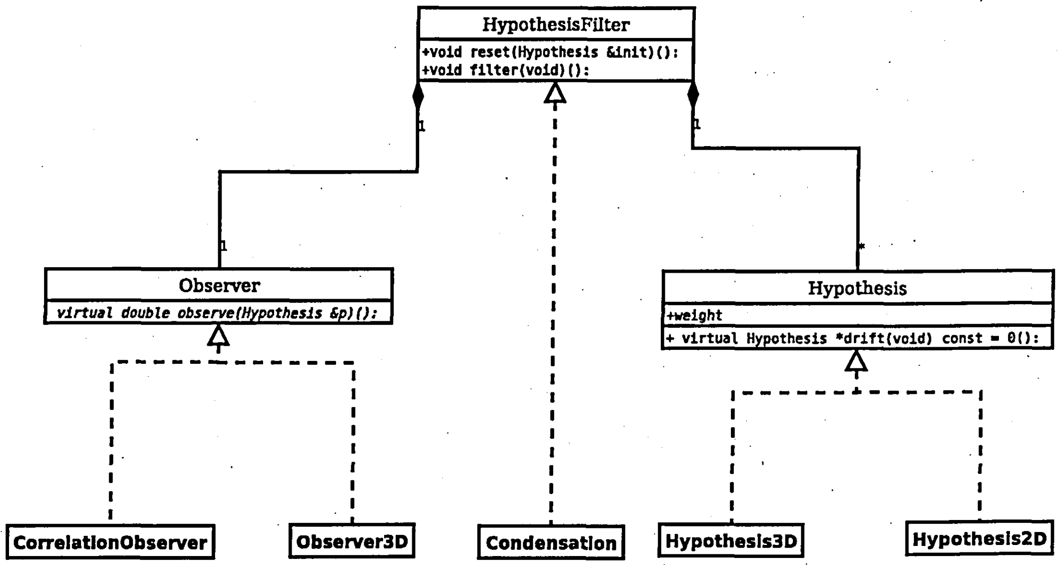
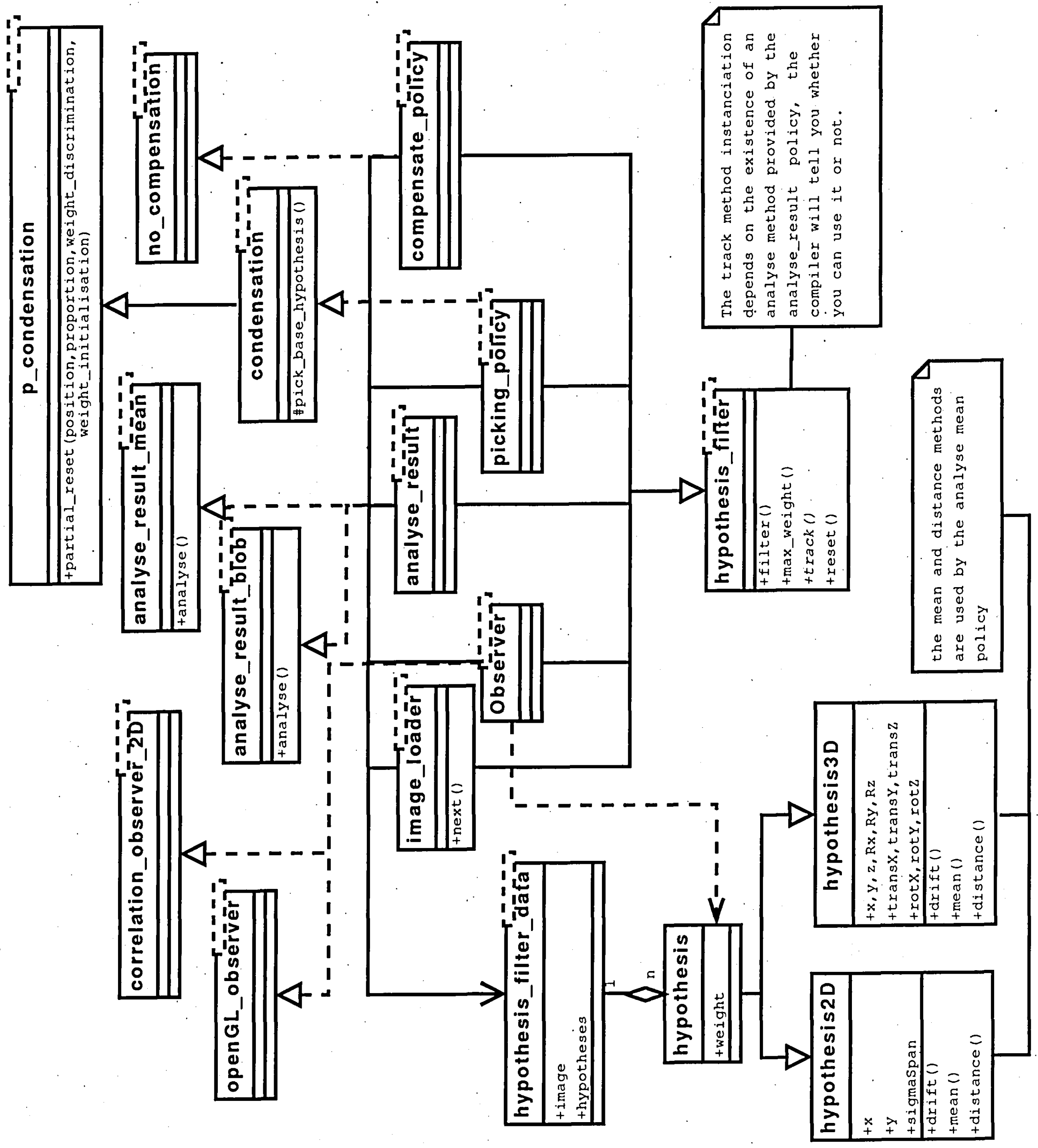


Figure 3.13: UML diagram of the first design of the particle filter



The track method instanciation depends on the existence of an analyse method provided by the analyse_result policy, the compiler will tell you whether you can use it or not.

the mean and distance methods are used by the analyse mean policy

Figure 3.14: UML diagram of the final implementation of the particle filter

3.7.4 Extensions

Parallelisation

Particle filter algorithms are well suited for parallelisation. The critical part of the code consists in evaluating the particle weights, this load can easily be shared amongst different processors by assigning to each one the evaluation of a number of particles. Therefore, the generalisation of multi-core processors provides another incentive to the usage of particle filters. Multi-threading can be used to implement this facility[93].

Detection based on particle filter

Adapting the particle filter for detection can be done easily. Inspired directly by Isard and Blake's paper on the ICONDENSATION[94] algorithm, the following two-stage-methodology is proposed. Using a coarse and fast measure the sampling of the image can be determined. From this sampling, refined measurements can be made to evaluate an object configuration. The main issue that remains is to determine a fast measure, which is a problem since conditions vary widely from one application to another. However, a relatively generic solution may involve the usage of colours.

3.8 Summary

In this chapter we have presented different methodologies, namely particle clustering, partial re-initialisation, integration of the dynamic of the system by over-weighting compatible particle weights, to track a translucent micro-pipette in a cluttered scene environment. The formulated procedures can be used to cope with problems arising from object occlusion and background clutter. Furthermore, by greatly reducing the number of measures that have to be taken after the first filtering stage, the particle clustering procedure allows the usage of more complex and precise matching method on a second filtering stage without significantly penalising the speed of the application.

The presented tracking algorithm, the particle filter, is robust, however the real-time objective remains an issue. This is due to the fact that the measuring methods to localise the object are time consuming. The tracking method *per se* is very efficient and allows tracking objects in high dimensional space. Future work can be carried out to determine some fast, highly discriminant measures. However, these measures are very dependent on the tracking scenario and ad-hoc solutions need to be provided for each case.

3.8.1 Possible improvements By first using a coarse filtering stage to determine the sampling of the state space, particle filters can be adapted for detection.

One of the major issues in tracking moving objects using template matching is that the original template becomes invalid due to the change in lighting conditions or due to the changes of the detected object features. To handle this situation, the model template could be updated. The simultaneous usage of different measures may give an indication when to update the template.

Depending on the number of processors available, parallelisation of the particle filters can be undertaken if the speed of the tracking algorithm only needs to be improved by a small multiplicative factor.

Chapter 4

State space, shape information and template reduction

4.1 Introduction

Objects or shapes are commonly represented by feature points. This chapter discusses how a subset of feature points representing a shape can still be robustly used for shape identification in an environment containing alternative shapes and in the presence of noise.

Two simple theoretical cases are discussed to introduce some of the issues that arise when trying to formalise and understand the nature of the problem of putting a shape into correspondence with another set of points. This issue arose in the field of computer vision while examining a broad range of variations of the Hough transform algorithm.

4.1.1 Square example Given a square, consider the number of points that are necessary to determine its characteristics, *e.g.* its position and size, without ambiguity. One point is insufficient since many squares have potentially this point in common as can be seen in figure 4.1.

Consider 2 points. Although it can be argued that 2 points provide more information, they are still insufficient to uniquely determine a square: if it is assumed that these 2 points belong to an edge of a square then there are many different squares of various sizes to which these 2 points can belong to. However, if we have the additional information that these two points are corners, then the number of possible squares intersecting these 2 points is finite and equal to 3, as the 2 points are either adjacent or on opposite corners. If they are adjacent there are 2 possibilities (see figure 4.2) and if they are opposite there is only one. It appears that knowing the characteristic of the points conveys some information of the location of the square. Note that, in this example, corners are not distinguishable so the four squares rotated by 90 degrees around their centre are considered to be the same.

Knowing the dimension of the square provides further information: if the square edge size measures a and if the distance between the 2 points is $\sqrt{2a}$ then there is only one possible square that fits these points, whereas there are still 2 possibilities when the distance between the 2 points is a.

Now, we shall assume that 3 points belonging to a square are known. What then can be inferred about the position of the square? If the 3 points are aligned they must belong to the same edge of the square. But there is insufficient information to determine the characteristics of the square since many squares of different sizes have these 3 points in common. We now consider the case when 3 points are not aligned. In appendix A.1 it is shown that for 3 unaligned points there exists an infinite number of squares that intersect these points. Figure 4.3 illustrates this for two specific three point configurations, that multiple squares can have 3 points in common.

Restating the problem, the objective is to identify the minimum number of points belonging to a square that uniquely identifies that square. In other words, we are looking for a set of points that intersects only one square, modulo the four 90-degree-rotations around its centre, the parameters of a square being its size and position. Note that the fact that we are looking for a square provides implicit information. This quantity of information varies with the shape that is being considered. The

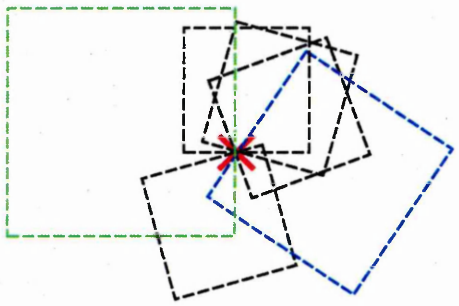


Figure 4.1: Squares sharing a common point.

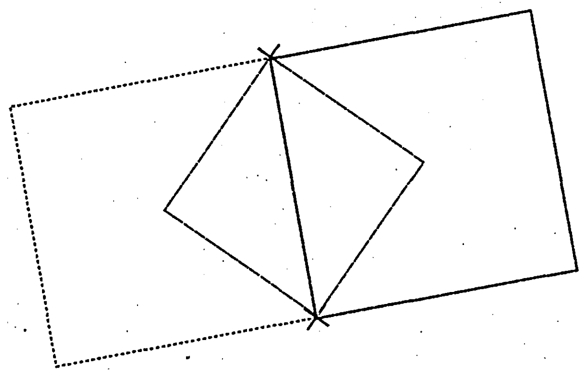


Figure 4.2: The three squares, modulo rotations, having these two points as corners.

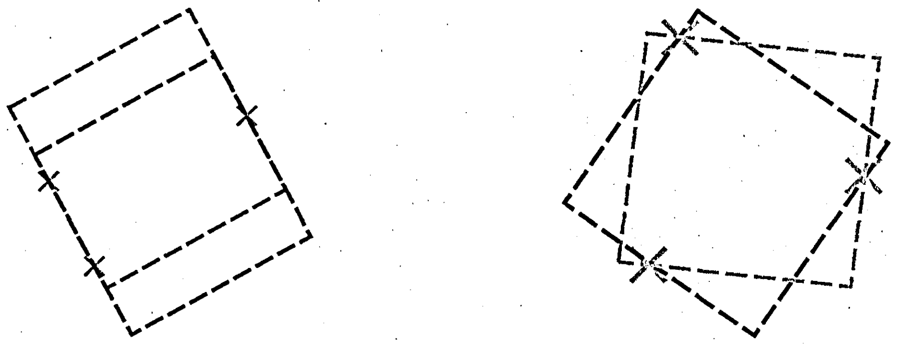


Figure 4.3: Two possible configurations of 3 points and example of squares that matches these configurations. On the left, 2 points are closer whereas on the right, points are equidistant.

more general problem of fitting any given shape to a set of points will be considered later. The previous cases demonstrate that some configuration of points will not provide sufficient information. For instance, n aligned points are not sufficient to uniquely determine the position of a square. However, configurations of 4 points exist that allow a square to be uniquely determined by these points. This is illustrated by figure 4.4. The 3 aligned points a,b and c must belong to the same edge of a

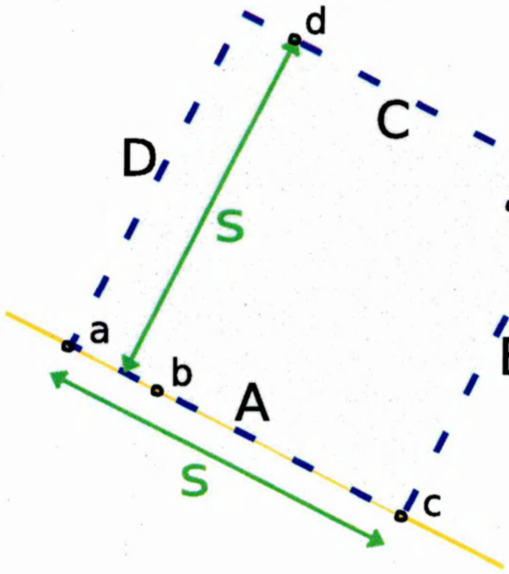


Figure 4.4: The points a,b,c and d uniquely define a square.

square, this edge is labelled A in the figure. The last point d must belong to the opposite edge of the square since its perpendicular projection on A belongs to the segment defined by the 3 other points, the edge of the square intersecting this point is labelled C in the figure. Moreover, the distance from d to A determines the size of the square. This distance must be bigger or equal to the distance between a and c in order for a square that passes through these four points to exist. Note that if a point belongs to the shaded area, like the point f, no square can fit the configuration of points a,b,c and f. However, if the distance between a point and the edge A is larger than the distance between a and c ($|ac|$), as illustrated by the point e, the configuration of points can be fitted by squares translated along the line passing through a and c, denoted (ac).

In general, it appears that some configurations of points intrinsically provide more information than others, even if the point configurations have the same number of points. In appendix A.2 we consider the problem of determining if there is a configuration of 4 points, when any 3 points extracted from the configuration are not aligned, that uniquely characterises a square. We do not solve the problem completely, however some ideas are provided to explore the problem in more depth. Note that many configurations of 4 points do not have any squares intersecting them. Another example is given in figure 4.5 (a). Figure 4.5 (b) and (c) illustrates that 4 points can belongs to different squares.

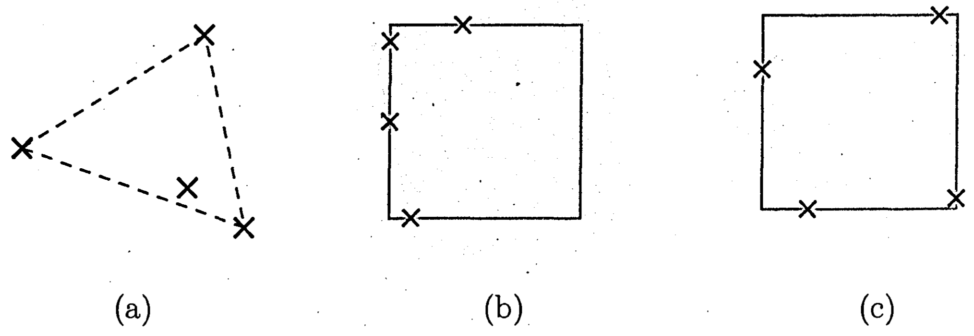


Figure 4.5: An impossible configuration and 2 possible configurations of 4 points that fit more than one square.

Figure 4.6 presents a configuration of five points that uniquely determines a square. In general, for a five point configuration, 2 points must belong to the same edge, therefore if no three points of the configuration are aligned, there exists at most a finite number of squares that have these five points in common; see appendix A.3 for further details. It appears that the more points are considered, the more likely the point configurations will not correspond to any square. More precisely, if a random number of n points are taken the probability that they do not to correspond to a square increases with the number of points. This can be understood by considering $n-1$ points of a configuration of n points. Only certain squares, if any, fit these $n-1$ points. The n^{th} point has to belong to one of these squares for the configuration of n points to be compatible with any square at all. Therefore, any additional points make the configuration more unlikely to be compatible with a square shape.

4.1.2 Circle example The case of a circle is now briefly discussed. One may consider that any group of 3 points contributes the same quantity of information since they specify the characteristic of a circle. The quantity of information is maximal since the position and size of the circle is completely known. If the diameter is known, 2 points at this given distance suffice to completely characterise a circle. If edge orientations of points are known, any 2 points are also sufficient to construct a circle. For an ellipse, 5 points are required.

Comparing the circle example and the square example, it appears that shapes convey implicit information about their position. This information vary according to the shape of the object and might be characterised or defined according to the point configurations that are needed to determine the state of a shape.

To sum up, when fitting a shape to a set of points some information is contributed by:

- the shape of the object.
- the transformation space, also referred to as the state space, associated with it. For instance, are only translations considered, or are rotations and scale required as well?
- the characteristics of the specified points, *e.g.* their colour, gradient direction, surrounding patch.

In this thesis we focus on a method that does not take into account the distinguishability of points. The methods, we are going to deal with, focus only on extracting the information from the relative position of points.

Nevertheless, these methods could be greatly optimised by taking advantage

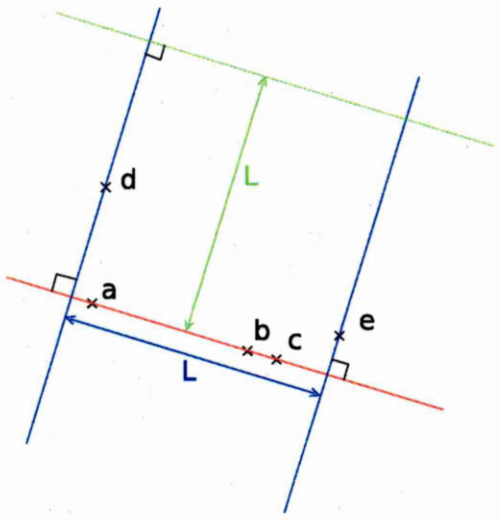


Figure 4.6: A configuration of 5 points that can be fitted by only one square.

of supplemental local information of points, such as their surrounding patch, that would considerably reduce the number of possible correspondences between points.

From the above discussion, it appears that the more points extracted from a given shape are considered, the more precisely this shape can be identified. Can this observation be quantified and in the case of a positive answer, can this be used to design more efficient algorithms? Also, how can configuration of points contributing more information than others be identified? For the last question two different perspectives can be adopted: points extracted directly from a shape and points extracted from a set of points to be fitted by a shape.

4.1.3 Locating a shape on an image Now consider an image that contains points that belong to a shape, for instance, a square, and points not associated to this shape. Since it is known that a square can be fitted in this set of points, if a set of n points, where n does not need to be very high, *e.g.* 10 points, are compatible with a square shape, it is very likely that the n points belong to the square and that its location is determined from these points. Of course if these points are aligned then multiple partially overlapping squares are compatible. To avoid this problem, the configuration of n points should be such that the points belong to different edges. A condition on the distance of the points, for instance, would ensure that the points belong to different edges. Similarly a condition of no more than p aligned points would ensure that the points belong to different edges too. Additionally, the number of points of the image should not be too high and densly positioned else the probability of configuration of points not belonging to the shape but compatible with it rises. This is usually not the case in real world applications.

What are the properties of a shape that can be used to characterise them? For the square we have been using the fact that edges are perpendicular and of the same size. If a shape is not a square, what criteria can be used to determine a set of points that uniquely characterises the shape? Given a set of points, is it possible to quantify the likelihood of the position of the shape that matches the points? In other words, given a set of points, is it possible to evaluate the probability density function (*pdf*) of the characteristics (*e.g.* its position) of the shape? Can points be determined to be part of the random set of points or of the shape? Is it possible to quantify the information contributed by an individual point? Is it possible to evaluate the additional information an additional point can contribute to a set of points?

Given a few conditions, all these questions can be answered for any shape and any 2-D transformations. In order to answer these questions the following abstract objects are defined:

- the shape S consists of a set of points,
- the transformation space T is another set representing the possible transformation that the shape can undergo.
- $p_i \in I$, $i = 1 \ldots n$, is a set of n points belonging to the image $I \subset \mathbb{R}^2$
- other prior knowledge (colour, edge orientation, moment of the surrounding patch *etc.*) can be represented by a mapping of the set of points S into a property space. However we consider that all points are identical and thus this possibility is not considered here.

Here, a shape is considered to be a set of points, but the previous examples used shapes that can be defined by their geometric properties. To define a shape as a set of points allows the matching algorithm to work with a wider variety of objects of which the shape may not be easily described in term of geometric properties. This allows the shape position to be parameterised independently of its nature. Another common approach used to describe a shape is by continuous contours; however, the discrete approach has many advantages over this approach, one of its major advantages is being able to take into account interior points, which bring significant information about the object, when it is partially occluded.

Considering a set of points in the image space we would like to, somehow, quantify the number of transformations that relate points of the shape with this set of points. The smaller the number of compatible transformations, the more information the set of points is contributing. Formally, we consider the set $\{t | \forall p_i \in I, \exists s_j \in s \subset S, t(s_j) = p_i, t \in T\}$.

So consider that the transformation space is the set of translations of $\mathbb{R}^2$ and the shape S to be a square of a given size. Then for a given point p of I, the set of transformations that correspond a point of S to p, if the space of translations were represented on an orthonormal plane, would look like the square S. Similarly for when another point q of I is considered.

Now if we consider both points at the same time we have to consider the transformations that put two points of S in correspondence with p and q. These transformations are the intersection of the set of transformations that put a point of S in correspondence with p and the set of transformations that put a point of S in correspondence with q. This set may:

- be the empty set ($\emptyset$), when the two points are too far apart.
- contain exactly one solution if the two points correspond to opposite corners of the square.
- contain 2 solutions when the 2 squares of the transformation space intersect in 2 points.
- contain an infinite number of solutions, when the 2 squares of the transformation space have their edges intersecting. This happens if the 2 points are horizontally or vertically aligned and are close enough.

This shift in perspective, in which the transformation space is considered, is the classical one that is used by Hough transform related algorithms. A point on the image space is selected and according to the transformation space and the shape under consideration all possible transformations that are compatible with corresponding a point of the shape with the image point will get one vote. After considering a certain number of points in the image space the transformations that obtain a similar number of votes are transformations that bring the points of S in correspondence with almost all the points of the image that were considered.

According to the nature of the shape and the transformation space, the set of transformations that are compatible with placing points of a shape in correspondence with a set of points of $\mathbb{R}^2$ might be discrete points or manifolds of the transformation space. Explicitly, if we consider the topology of this set it might consist of isolated point, surfaces or volumes of different dimensions, lower than, or equal to the dimensions of the transformation space.

Knowing the shape we are looking for, and the space of the transformations that describes the modifications by which the shape can be affected, we would like to evaluate the quantity of information that is contributed by a set of points of the image. However, the set of transformations that are compatible with the points of the image under consideration might have different topological elements. For instance, a set may consist of a few isolated points and a line of points of length l. Assuming that the transformation space is of dimension 2 and that the quantity of information is given by info(l), info() being a decreasing function of the number of compatible transformations with the image points, then isolated points would possibly not be taken into account since the length of an isolated point is null. This is problematic since these points may contain the position of the shape and contribute a significant amount of information.

To avoid this issue balls of radius δ are considered. Where δ corresponds to the accuracy with which the transformation is known. The dimensions of the balls (hyperballs) correspond to those of the the transformation space. By considering the minimal number of balls that can cover the set of transformations we obtain a number. The smaller this number, the better the shape state is known. If this number is equal to 1, one may consider that the shape transformation is known with sufficient accuracy.

Definition 1 *Let T be the transformation space, $\mathrm{B}_\delta(p)$ an open ball of the same dimension than T, of radius δ and of centre p. P a set of points of the image, S the set of points of the shape, $C(P)$ the set of transformations compatible with the points of the image. We define the quantity of transformations $\mathcal{Q}_\delta^{T,S}(P)$ contained in $C(P)$ as:*

$$C(P) := \{t \in T | \mathrm{card}(t(S) \cap P) = \mathrm{card}(P)\}$$

$$\mathcal{Q}_\delta^{T,S}(P) := \min \mathrm{card}(\{\mathrm{B}_\delta(p) | p \in T, \delta \in \mathbb{R}, \mathrm{B}_\delta(p) \cap C(P) \neq \emptyset\}) \tag{4.1}$$

A set of points P is said to uniquely characterise a shape transformation relative to a transformation space T and with an error δ if $\mathcal{Q}_\delta^{T,S}(P)$ is equal to 1.

In practice, when using images, a discrete bounded 2-D plane is considered. This plane is divided into cells or pixels (picture elements) so the position of a point in this space is approximated by the pixel encompassing it. Therefore, we deal with a small 2-D surface instead of a single point. Similarly a template is, in practice, given as a set of pixels. Considering that pixels are a set of points, in order not to miss any possible compatible transformation, one should take into account all transformations where the projection of a template pixel intersects an image pixel that is a feature point. As a consequence, if we consider that T is the translation space, the template is a pixel and the image contains one pixel as a feature, the set of transformations compatible is a 2-D surface of the translation space.

We now consider a set of points S' extracted from a shape and we consider that T is bounded. The following quantifies the information contributed by a set of points of a shape relative to a transformation space T and an accuracy of δ.

$$\mathcal{I}_{\delta}^{T,S}(S') := \max_{t \in T} \mathcal{Q}_{\delta}^{T,S}(t(S')) \tag{4.2}$$

We call $\mathcal{I}_{\delta}^{T,S}(S')$ the characterising value of a set of points relative to a shape and a transformation space. When the characterising value is equal to 1 it means that for any transformation of the transformation space of this set of points the position of the shape is characterised uniquely and completely, in that sense the information brought by the subset of points is maximal. If the value is n it means that for a given position of the transformation space the set of points is compatible with n positions that are distant enough to be distinguished. Note that finding a set of points in an image, that characterises uniquely the position of a shape relatively to a transformation space, does not mean that the shape is necessarily present since the points may have taken this shape by coincidence therefore a verification stage is needed, however if the size of a characterising set of points for a shape is chosen appropriately the number of verifications should be small.

When selecting a subset of points from a shape, the number of configurations to consider quickly becomes large: for k point configurations out of n points of the shape the number of cases to consider is $\binom{n}{k}$. Arguably, only small sets of points are of interest in representing the object and therefore k should be small, *e.g.* 10 to 40 points, but the possible number of configuration that could represent the object may be quite large. Monte Carlo methods and genetic algorithms are good candidates to find configurations that are close to optimal. Details as to how to to evaluate $\mathcal{I}(S')$ are given in section 4.5.

4.2 Self similar set of points

Thus far, the state of a shape has been characterised with a limited set of its points extracted from an image. Another fecund perspective to examine this question is to consider a group of p points from the shape and to consider where else these points can fit the shape relative to a set of transformations. When the only transformation that allows the points to fit the shape is the identity, then the configuration is considered to be characteristic of the shape and will be qualified as such. If there exists transformations that map the set of points to another set of points of the shape then this set of points is said to be self similar. Two examples are given here:

1. We consider the problem of identifying a template image in a larger image. If points can be uniquely identified, for instance using their surrounding pixel values, and if only translations are considered, it suffices to match a point in the larger image with the template image to determine the position of the template image. Once a point of the template image is located, all other point positions are known.

   Nevertheless this is a different problem to the one we have been exploring up to now since we have been considering shapes consisting of indistinguishable points. Using distinguishable points requires less points to characterise an object but, as a trade-off, more computation is needed to compare them. However, the techniques discussed can be adapted to the case of distinguishable points. Their advantages in terms of reducing the template size might be less dramatic since templates using distinguishable points tend to be significantly smaller. A balance has to be found between the degree of distinguishability of points, its computational cost and the size of the set of characterising points needed to identify the state of a shape.

2. Considering again points with unknown associations; one point cannot characterise a shape since it could be any point of the shape. However, if there is a set of two points having a unique configuration in the shape, *i.e.* if the vector from one point to the other is different from any other vector between 2 points of the shape, then finding 2 points with the same configuration on an image that uniquely contains the shape allows its position to be characterised.

   Note that when the transformation space consists of translations, any shape made of more than 2 points does have such a unique configuration of 2 points. Consider 2 points of the shape that are furthest apart. Assuming that these 2 points have a self similar configuration of points relative to a translation then there would exist 2 points that are further apart than these 2 points. Indeed, the 2 points on a diagonal of the quadrilateral which corners would be the 4 above mentioned points would be further apart than the 2 points furthest apart. This would be absurd and proves the existence of such a configuration. For instance, a circle has a infinite number of such configuration of points: points diametrically opposed. None are self similar relative to translations and all of the are characteristic of the circle.

This also leads to an algorithm to identify the translation of an object: First of all, the shape, a set of points extracted from a template image is considered. Since we want the algorithm to be robust to noise, matching points should be redundant. This avoids false matching when background points are present or feature points of the object are not detected. As a consequence the template is not reduced to 2 points furthest apart but to a few pairs of points characterising the shape. The data structures that are used to store the point associations and the image feature points affect the speed of the algorithm. Then each feature of the image is alternatively considered to be a point of the reduced shape, the point associations are probed and the points that account for most of the association are considered as good candidates to be points of the reduced shape. While checking for the associations the process can be stopped earlier in case of an apparent mismatch. Other optimisation schemes could be devised here. To guarantee that the obtained match is correct a verification stage can be added.

The number of feature points in the image is critical. As a consequence the choice of the feature detector is also critical. Indeed it should not be much slower than the shape locator and it should also be selective enough to have as few points as possible so that the location of the shape can be done efficiently. This remark might seem trivial but it comes out of experience in trying to deal with a large amount of features because a feature detector with poor discrimination was used (in our case a Canny edge detector). This yielded tens of thousands of features for an image.

If the image is filtered with a selective feature detector and has fewer points it leads to an efficient recognition algorithm.

Note that points that are far apart relative to the size of the object are more likely not to be in a self similar configuration since there are fewer equivalent configurations in the object. Any other criteria that would reduce the number of possible configuration of points from the whole set of points might have this beneficial property. It is possible to find counter shape examples where points far apart have self similar configurations, but since the number of such point configurations is limited compared with all possible configurations, the likelihood of having self similar configurations relative to a translation is smaller than taking any point configuration from the shape.

Finding a criteria to determine a set of points for transformation spaces that are different from the translation space does not seem trivial. Moreover since any transformation space has its own characteristic, such a criteria should be sufficiently generic. This is the problem that we tackle in the next section.

Note that the shapes that are considered should have many more than a few features, *e.g.* hundreds or more rather than say tens.

4.3 Robustness to noise

We reiterate our objective: to reduce a shape template to a few points that characterise the shape and to identify it in a larger image more efficiently. One issue with real images is that some points belonging to the shape will be missing while many other points from the background might disrupt the detection of the shape by creating false positives. Also, additional points from the object underlying the shape might appear due to illumination changes or even sensor noise.

To cope with this issue a minimal set of points characterising the shape is not sufficient. Indeed, if a point of the configuration does not appear or, less probably, if a set of points, not all belonging to the object characterised by the shape, matches a minimal characteristic configuration of the shape, the shape position will not be detected correctly.

One way to work around this issue is to consider characterising point sets that remain characterising point sets even if one of their points is removed. Thus a margin of error of one missing point is obtained. Similarly, characterising sets of points might be built with a larger margin of error. Note that to construct a characterising set of points with a margin of error of one missing point, it may be necessary to add multiple points to a minimal characterising set of points. Also, each point added to a characterising set of points decreases the possibility of a match with a random set of points. It should also be noted that the more points are used to represent the object, the more likely it is that some of them may be missing.

4.4 Template reduction, a simple example

Let us consider the shape S that consists of 7 pixels with its associated transformation space T that consists of horizontal translations in the range -2 pixels 2 pixels that are represented in figure 4.7. L represents the look-up table where colours correspond to one or multiple transformations. There are 2 possible reduced templates that consists of 1 pixel. They are shown in figures 4.8 A and B. There are multiple possible reduced templates of 2 pixels, figures C and D are two examples. As for figure 4.8 E and F they show two ambiguous templates that do not systematically refer to a unique transformation. By observing the look-up table (L) in figure 4.7 it can be seen that if one of these reduced templates was chosen, in the case where the blue and black shaded pixel and the purple and black shaded pixel of the look-up table were selected, it would not be possible to know if the transformation was a translation by 1 or -2 pixels. Note that these 2 reduced templates are the only one that are self similar relative to the transformation space (state space). Any reduced templates with 3 pixels or more uniquely characterises the shape. Indeed none are self similar; a systematical way to verifying this consists of using the look-up table and for each transformation of the state space to check that the look-up table provides a unique transformation; which is necessarily the correct one.

4.5 Evaluation of the characterising value of a set of points

A methodology to evaluate the characterising value of a set of points P from an image is hereby discussed. Some experimental data is presented in section 6.2.

A bounded transformation space T is divided into small hypercubes $h \in H$ such that H is a partition of T. The reason why we consider hypercubes is because they are easy to implement. However, the term "hyper-parallelepiped" would be more appropriate since the size for different dimensions may differ.

Take S as being the set of points of the shape that undergoes a transformation given by: $t(S) := \{t(s)|s \in S\}$. $l_{i,j}$ are the elements of L, which is a 2-D array of the size of the image, that correspond to the points overlapped by the pixel on line i and column j noted $I_{i,j}$.

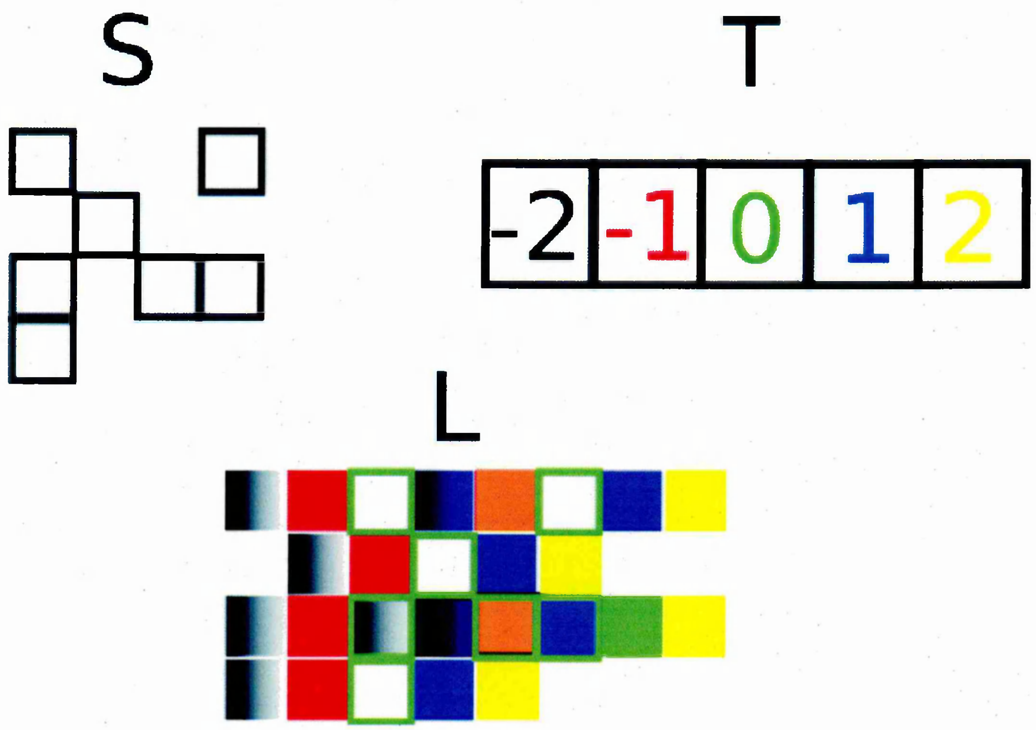


Figure 4.7: A template (S), its associated state space (T) and the corresponding look-up table (L). Colours are used to represent the states referenced by the look-up table.

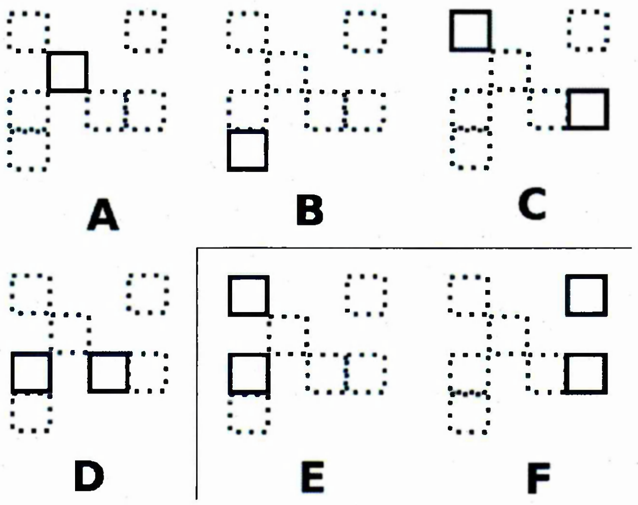


Figure 4.8: Figures A-D show non ambiguous reduced templates. Figures E,F show ambiguous reduced templates

For each set of points $h \in H$ and for each $t \in h, t(S)$ is evaluated and a reference to h is stored in $l_{i,j}$ whenever $I_{i,j} \cap t(S) \neq \emptyset$. This operation is not completely trivial to implement and needs some approximation to be done in a reasonable amount of time, but time is not critical since these operations can be performed offline. However, the amount of computation can be huge, especially when working with high-dimensional spaces in order, for instance, to take into account translations, rotations, change of scale, shearing and perspective transformations. Once this is completed, $l_{i,j}$ contains the references to the hypercubes that contain a transformation that projects a point of the shape to the pixel $I_{i,j}$. We note $h_{i,j}^k, k = 1..n_{\text{ref}}$ the hypercubes referenced in $l_{i,j}$.

It is now shown that the number of references n_{ref} that is contained by $l_{i,j}$ is equal to $f\mathcal{Q}_\delta^{T,S}(I_{i,j})$ with $a \leq f \leq b$ where $(a, b, f) \in \mathbb{R}^3$, a and b are two constants that depends on the size of the hypercubes and the error ball.

All hypercubes are assumed to have the same size but the demonstration holds with hypercubes of different size. In order to demonstrate it, one just has to consider the extreme cases. If the minimal number of balls needed to cover one hypercube is m then at worst the minimal number of balls to cover all hypercubes would be $m \cdot n_{\text{ref}}$ thus $\mathcal{Q}_\delta^{T,S}(I_{i,j}) \leq m \cdot n_{\text{ref}}$. If a ball can intersect at most p hypercubes then we need at least $\frac{n_{\text{ref}}}{p}$ balls to cover all transformations that are in the hypercubes and thus $\frac{n_{\text{ref}}}{p} \leq \mathcal{Q}_\delta^{T,S}(I_{i,j})$.

Lemma 1 *$\exists(a, b) \in \mathbb{R}_+^{*2}$, such that $a \cdot \mathcal{Q}_\delta^{T,S}(I_{i,j}) \leq n_{\text{ref}} \leq b \cdot \mathcal{Q}_\delta^{T,S}(I_{i,j})$*

The most unfavourable case being when a set of transformations, that can be covered by a unique ball, lie on the boundary of multiple hypercubes. If the dimension of the transformation space is n, the number of hypercubes that cover the set of transformations can be as high as 2^n. As a consequence, in practice, not only must n_{ref} be considered but also whether the hypercubes are contiguous or not.

Therefore, it is possible to have an evaluation of the quantity of the transformations compatible with a set of points P by considering:

$$\mathcal{Q}_\delta^{T,S}(P) \sim \text{card}\left(\bigcap_{i,j|I_{i,j} \in P} \bigcup_k h_{i,j}^k\right) \tag{4.3}$$

and as a consequence, it is also possible to evaluate the characterising value of a subset of points S' of the shape by considering:

$$\mathcal{I}_{\delta}^{T,S}(S') \sim \max_{t\in T} \mathrm{card}\Big(\bigcap_{i,j|I_{i,j}\cap t(S')\neq\emptyset} \bigcup_{k} h_{i,j}^{k} \Big) \tag{4.4}$$

which is much more computationally expensive to evaluate. In order to reduce computation time it is recommended that the transformation space is sampled. For instance by selecting a few transformations for each hypercubes, this may result in a good approximation. Proofs, experiments and more theoretical studies remain a future topic of research.

4.6 A generic algorithm for the pose estimation of rigid objects

Decimating the template shape and using it with a Hough transformation algorithm is equivalent to reducing the number of transformations that a point in the image space can vote for. Thus a 1 to n mapping is obtained with n being significantly smaller than it would have been before the template reduction. Additionally, the idea of the probabilistic Hough transform (PHT), which determines the state of an object by randomly selecting a subset of the points in the image space, can be used. On top of which, using the idea from the randomised Hough transform (RHT) which consists of a n to 1 mapping from the image space to the parameter space, the simultaneous selection of a few points P in the image space reduces further the number of transformations compatible with these points ($\mathcal{Q}_{\delta}^{T,S}(P)$) resulting in a few to few mapping. Algorithm 1 combines these three ideas. These three ideas aim at reducing the number of computation to determine the pose of an object. The decimation of the template does this by studying the shape of the object, the PHT by considering only part of the points from the image space, which, for tracking purposes, as most of the points considered in a region of interest belong to the tracked object, should perform well, and the RHT by limiting the number of votes to cast in the parameter space. In the next chapter, it will be shown that the usage of a lookup table provides yet another way to reduce the amount of operations needed to find the position of an object.

Examples of criteria that can be used in stage 2 are: when the transformation space is distant invariant, are the distances between the selected points compatible

Algorithm 1: Outline of the proposed methodology to locate a shape in an image using features.

```
Stage 1: Select $p$ feature points from the image space.
Stage 2: Use different criteria to check if the feature points are compatible
with the object underlying the shape. If it is compatible go to stage 3 else go
back to stage 1.
Stage 3: Evaluate the transformations that are compatible with these
features and vote for them. The smaller the number of transformations the
more discriminative is this stage.
Stage 4: If one of the transformation that just get a vote on the previous
stage received enough vote go to stage 5 else continue with stage 1.
Stage 5: Verification stage. If the shape has not been identified correctly
remove the votes for this transformation and go back to stage 1. Else return
the object state.
```

with the distances of the points of the decimated template? When the transformation space is not distant invariant because scale, skew or projective transformations have to be taken into account, a non geometric criteria such as the colour of the features can be used. When scale is the only non distant invariant transformation that is taken into account, the curvature of connected features may also be used. Stage 2 is critical since it can reduce considerably the computation of the algorithm. Moreover, the number of irrelevant transformations of the object location are thus reduced significantly, which results in a more robust algorithm. Alternatively, stage 1 and 2 can be merged by selecting features according to one or more criteria.

The number of randomly selected features in stage 1 depends on the proportion of features belonging to the object relative to the total number of features, how much the template shape has been decimated and the transformation space under consideration. Currently, this number has to be determined experimentally for each given application. When the previous location of the shape is known features can be selected in a constrained area and are more likely to belong to the shape, which subsequently can greatly reduce the number of operations needed for the determination of the shape position.

Similarly, the number of votes in stage 4, that is judged to be large enough to determine that a transformation is good enough to be verified, has to be determined experimentally since it depends on the proportion of feature points that does not belong to the shape and that could be compatible with a transformation of the shape. However this number should not be greater than $\binom{n}{p}$, n being the number of object features and p being the number of randomly selected feature points at stage 1, otherwise it would mean that the same configuration of points belonging to the shape have to be selected more than once. In practice, this number is much smaller.

Often stage 5 is omitted as the result from stage 4 is considered to be good enough. This allows a simpler algorithm implementation and arguably a more robust one, however the last stage may allow a better guarantee of the result than it would be possible with the technique used in stage 4. Moreover, it allows more flexibility, and depending upon the efficiency of the technique chosen for the verification stage, a speed up might result with the right balance between the two stages. Indeed, the number of votes necessary to decide on a transformation verification may be decreased in such a way that the object transformation is found earlier.

The next chapter explains how stage 3 can be performed efficiently when an approximation of the position of the object is known, for instance, when the motion of the object is bounded and its previous position is known.

4.7 Summary

Considering a set of points characterising a shape and a transformation space, the issue of selecting a subset of the points that can still characterises the original shape has been discussed and developed.

Given a set of points, a shape and a transformation space the size of the set of transformations compatible with matching the shape to the set of points was considered and used to define the characterising value of a subset of points.

The relationship between self similarity between a set of points relative to a transformation space and a characterising set of points of a shape was then explored. The issue of the robustness of a characterising set of points to randomly positioned points was mentioned and a definition for the margin of error of a characterising set of point was proposed.

A practical algorithm based on the Hough transform was then developed to measure the characterising value of a subset of points of a shape for a given bounded transformation space. The equivalence of the result provided by the modified Hough transform and our definition of the quantity of compatible transformations was derived. This algorithm can be used to reduce the number of points of a template such that recognition and tracking algorithms' speed performances can be improved without significantly impairing the robustness of the template.

Finally a methodology to estimate the pose of a rigid object was proposed and discussed. The focus of the discussion was on different optimisation technique to reduce the number of operations needed to determine the position of an object.

4.7.1 Future research It is possible to evaluate the reduced template for some unbounded transformation spaces such as rotations and translations. It is sufficient to consider a shape covered by all its possible translations and rotations. Since far away translations does not intersect the shape and the set of rotations is limited the same methods used for the bounded case can be employed.

For recognition, future work could include the development of a characteristic set of points relative to a set of shapes and not a unique shape. The outcome of defining, if possible, a scalable methodology to iteratively construct characteristic sets of points of a shape when an additional shape is added to a set of shapes will have significant consequences.

Finally, it should be explored if the concept of entropy which is linked with the concept of information could be used to determine the size of the minimum number of points needed to characterise a shape relative to a state space.

Chapter 5

The stencil estimator

5.1 Introduction

Inspired by [6] [95] [96], the use of stencils is proposed to estimate the position of a shape, *i.e.* a set of points, in an image. It is shown that this estimator can be efficiently evaluated using a variation of the Hough transform. In this chapter italicised capital letters shall be used to represent sets.

5.2 The stencil estimator

The bounded transformation space T, which is the set of transformations a shape can undergo, is partitioned into n subsets of possibly different sizes. In practice, to ease the implementation, the subsets are "hyper-rectangles" that partition T. Each subset $U \subset T$ is associated with what we refer to as a stencil $\mathrm{S}_{U,O}$ which is the set of positions of the model features O (also referred to as the template or the shape) occupied when the shape is moved according to the elements of the transformation subset U.

$$\mathrm{S}_{U,O} = \{t(p) : t \in U, p \in O\}$$

In the remainder of this chapter, when there is no ambiguity, S_U shall be used in place of $\mathrm{S}_{U,O}$. These notations are illustrated by figure 5.1.

We define:

$$\begin{array}{rccc} \Psi_O : & \mathcal{P}(T) & \rightarrow & \mathcal{P}(\mathbb{R}^2) \\ & U & \mapsto & \mathrm{S}_{U,O} \end{array} \tag{5.1}$$

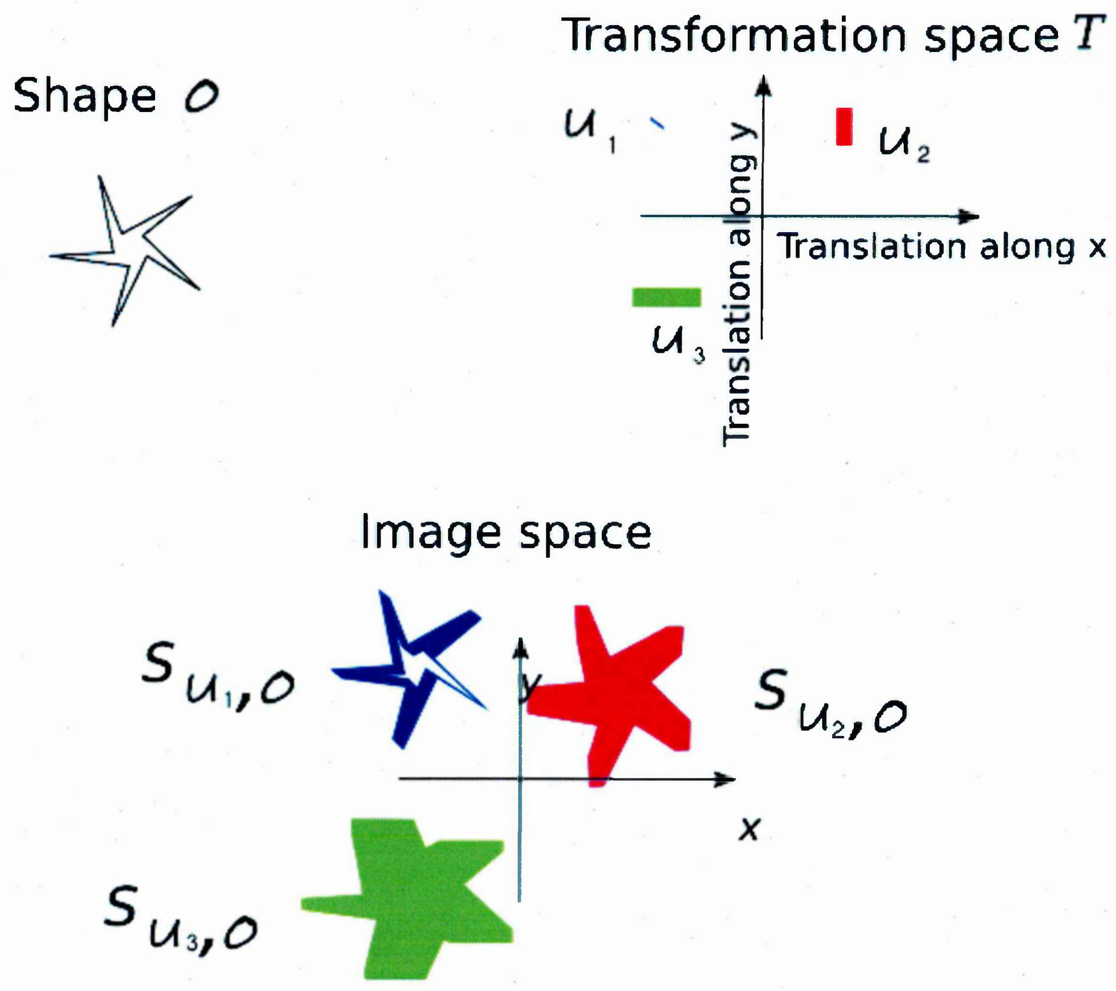


Figure 5.1: The stencils of the "star" shape corresponding to the coloured area of the transformation space have been drawn in the image space. The coloured regions are the set of points overlapped by the shape when it is moved according to the transformations of the subset having the same colour in the transformation space.

which is the function that associates a transformation subset to its corresponding stencil. $\mathcal{P}(E)$ is the set of all subset of E.

Given a set of n points $P = \{(x_i, y_i)\}_{i=1}^{n} \in \mathbb{R}^2$ the stencil estimator is defined to identify the set of the stencils that intersect the maximum number of points of P.

Definition 2 *Let M be a* partition *of T, $U \in M$, O the object points and P a set of points of $\mathbb{R}^2$. The Stencil Estimator (*SE*) is defined as:*

$$\mathrm{SE}_{M,O}(P) := \arg\max_{U \in M} \mathrm{card}(p \in \mathrm{S}_{U,O} : p \in P) \tag{5.2}$$

$\mathrm{SE}_{M,O}(\cdot)$ is the estimator used to determine the position of the shape. The remainder of this section expands on a few properties of this estimator.

For a given point p, the span of a transformation set is defined as the maximum distance between the resulting transformed points.

$$\mathrm{span}_p(U) := \max_{s,t \in U} \mathrm{dist}(s(p), t(p)) \tag{5.3}$$

where $\mathrm{dist}(\cdot,\cdot)$ is the Euclidean distance. The subscript p is omitted when the span is independent of the object points; this is the case for translations. For a set of points O, we denote $\mathrm{span}_O(U)$ as the maximum span over O:

$$\mathrm{span}_O(U) := \max_{p \in O} \mathrm{span}_p(U) \tag{5.4}$$

We also use the following notations:

- $\forall U \in M, U_p := \{t(p) : t \in U\} = \Psi_{\{p\}}(U)$
- $\forall t \in T, t(O) := \{t(p) : p \in O\}$

Note that with these notations $\mathrm{S}_U = \bigcup_{p \in O} U_p$

Lemma 2 *If $\forall p, q \in O$ and $\forall U, V \in M$ we have $U_p \cap V_q = \varnothing$ with $U \neq V$ then*

$$\mathrm{card}(p \in \mathrm{S}_U : p \in t(O)) = \begin{cases} \mathrm{card}(O) & \textit{if } t \in U \\ 0 & \textit{otherwise} \end{cases}$$

Demonstration: $t \in U$, thus $t(O) \in \bigcup_{p \in O} U_p = \mathrm{S}_U$ it follows that $\mathrm{card}(p \in \mathrm{S}_U : p \in t(O)) = \mathrm{card}(O)$ if $t \in U$.

If $V \neq U, \mathrm{S}_V \cap \mathrm{S}_U = (\bigcup_{p \in O} V_p) \bigcap (\bigcup_{p \in O} U_p)$ thus $\mathrm{S}_V \cap \mathrm{S}_U = \bigcup_{p,q \in O}(V_p \cap U_q) = \varnothing$ and therefore $\mathrm{card}(p \in \mathrm{S}_V : p \in t(O)) = 0$ if $t \in U$ ♦

According to this lemma, a necessary condition for a stencil S_V to receive votes from $t(O)$ with $t \in U \neq V$ is that $U_p \cap V_q \neq \varnothing$. This specific case may occur when the object's points are "sparse" enough and the partition of the bounded transformation space thin enough so that the stencil of a point does not overlap the stencil of any other point. Moreover, for a given point p, the condition $U_p \cap V_p = \varnothing$ is true only if translations are considered but becomes false when additional degrees of freedom like scale transformations or rotations are considered.

Lemma 3 *If $t \in T$, a set of transformations that is distance invariant, and if*
$\forall U \in M, \mathrm{span}_O(U) < \min_{p,q \in O, p \neq q} \mathrm{dist}(p,q)$
then $\forall U \in M, \forall p \in O, \mathrm{card}(t(O) \cap U_p) \leq 1$

Demonstration: If $\exists q \in O$ such that $t(q) \in U_p$ then $\forall r \neq q \in O$ we have:
$\mathrm{span}_p(U) \leq \mathrm{span}_O(U) < \mathrm{dist}(q,r) = \mathrm{dist}(t(q), t(r))$
Thus, $t(r) \notin U_p$ ♦

In other words, under these conditions the stencil of a point can receive at most one vote from $t(O)$.

Theorem 1 *If T is a set of translations of $\mathbb{R}^2$, if $t \in U$ and*
if $\forall V \in M, \mathrm{span}_O(V) < \min_{p,q \in O, p \neq q} \mathrm{dist}(p,q)$
then $\mathrm{SE}_{M,O}(t(O))$ is unique and equal to U

Demonstration: $U \in \mathrm{SE}_{M,O}(t(O))$ is trivial since $\mathrm{card}(t(O) \cap \mathrm{S}_U) = \mathrm{card}(O)$ which is the maximum number of votes a stencil can get.

It will now be shown that all other stencils will obtain fewer votes. First consider the case $\mathrm{card}(O) = 1$ then if $t \in U, t(p) \in U_p$ by definition and $t(p) \notin V_p$ because we are in the case of a translation and $U_p \cap V_p = \varnothing$

We now consider the case $\mathrm{card}(O) = 2, O = \{p, q\}$. We assume that $\exists V \neq U \in M$ such that $\mathrm{S}_V \cap t(O) = 2$ because $U_p \cap V_p = \varnothing$ it means that $t(p) \in V_q$ and $t(q) \in V_p$ which means that $\exists t_1, t_2 \in V$ such that $\begin{cases} t_1(p) = t(q) \\ t_2(q) = t(p) \end{cases}$ Since we are only considering translations this can be written: $\begin{cases} t_1 + p = t + q \\ t_2 + q = t + p \end{cases}$

$\Rightarrow 2(p-q) = t_2 - t_1 \Rightarrow 2\|p-q\| = \|t_1 - t_2\| < \text{span}(V)$ which is absurd since $\|p-q\| > \text{span}(V)$.

Now consider the general case: $\text{card}(O) = n, n > 2 \in \mathbb{N}, O = \{p_i\}_{i=1}^{n}$. We are going to show that it is inconsistent to assume that $\exists V \neq U \in M$ such that $\text{card}(\mathrm{S}_V \cap t(O)) = \text{card}(O)$ with the previous hypotheses. According to lemma 3, $\text{card}(V_p \cap t(O)) \leq 1$ thus it follows that $\forall p \in O, \text{card}(V_p \cap t(O)) = 1$. Since $U_p \cap V_p = \varnothing$ $\forall p_i, p_j \in O, i \neq j, \exists t_{ij} \in V, t_{ij}(p_i) = t(p_j)$ in other words, this hypothesis implies that there is a set of translations belonging to V that generates a permutation without fixed points between O and $t(O)$. Thus, there exists a cycle of size m comprised between 2 and card(O) such that:

$$\begin{cases} t_{ij}(p_i) = t(p_j) & (1) \\ t_{ki}(p_k) = t(p_i) & (2) \\ \dots & \\ t_{j\alpha}(p_j) = t(p_\alpha) & (m) \end{cases}$$

By multiplying the first equality by $m-1$ and subtracting all other equalities we obtain: $t_{ij} - t_{ki} + \cdots + t_{ij} - t_{\alpha j} = m(p_j - p_i) \Rightarrow m\|p_j - p_i\| \leq (m-1)\ \text{span}(V)$. Which is absurd since $m\|p_j - p_i\| > m\ \text{span}(V)$ and thus proves the uniqueness of the solution ♦

The practical results of this theorem are limited since, in reality the accuracy with which the features are located depends on the selected feature detector and there are a number of false positives and false negatives due to various reasons (*e.g.* discretisation, background objects, occlusion, accuracy of the feature detector). Moreover, more complex transformations than translations are often of interest.

Note that if transformation spaces with more degrees of freedom are considered, in order to take into account rotations or scale changes for instance, the property $U_p \cap V_p$ does not hold any longer and the stencil estimator might indicate multiple sets. However, for rotations, as tests have indicated, this does not seem to be a big problem. The choice of partitioning T into hypercubes, which seemed easier to implement, might have beneficial properties regarding the robustness of the estimator. This result reinforces the intuition that having sparse features reduces the feature sets compatible with a stencil as mentioned in [6]. Also, the estimator can be used to give a first approximation of the position of the object. Having multiple answers is thus not a big issue.

5.3 Robustness

Some aspects of robustness have already been discussed in section 4.3. In order to quantify robustness and characterise how well the stencil tracking can cope with disturbances, a few definitions are proposed. The ideal case, where only object features are present, differs from real image data. These differences are often classified into two categories:

- false positives, the set of features that appear and that could not have been predicted knowing the position of the object. We denote them as P.
- false negatives, the set of features that would have been predicted knowing the position of the object and that are missing in the filtered image. We denote them as N.

Imperfections of the feature detector, which may be tuned to influence its rate of false positives and negatives, are not the only cause of false positives and false negatives. Indeed, since the object is not alone in the scene, background elements provide false positive features. The tracked object may also be occluded thus generating additional false positive and negative features.

Definition 3 *The margin of error (*ME*) of a stencil relative to a set of stencils is:*

$$\begin{aligned} \mathrm{ME}_M(\mathrm{S}_{U,O}) &:= \min_{P,N,t\in U} (\mathrm{card}(P) + \mathrm{card}(N)) : \\ &\exists V \neq U \in M, V \in \mathrm{SE}_{M,O}((t(O) - N) \cup P) \end{aligned} \tag{5.5}$$

with M being a partition of the transformation T, U and V elements of M and P and N sets of points of $\mathbb{N}^2$

For a given stencil, $\mathrm{S}_{U,O}$, its margin of error is defined such that, for any transformation $t \in U$, the margin of error represents the minimum number of feature to remove from $t(O)$ and to add to the image such that another stencil obtains the same number of votes. In other words, this defines the minimum number of errors that might trigger a different stencil than the desired one.

Definition 4 *The overall margin of error* ME *of a stencil estimator is:*

$$\mathrm{ME}(\mathrm{SE}_{M,O}) := \min_{U\in M} \mathrm{ME}_M(\mathrm{S}_U) \tag{5.6}$$

This represents the minimum margin of error of all stencils of the stencil estimator. In other words, this defines the minimum number of errors that might trigger an undesirable answer from the stencil estimator. A number of questions arise:

1. Is it possible to increase the margin of error as defined?
2. Does the margin of error accurately characterises robustness?
3. Is it possible to keep the same robustness while increasing the speed of the algorithm?
4. How does decimating the template influence the margin of error?

To analyse these questions we have undertaken a few tests, but before presenting the results, the implementation of the stencil estimator is first discussed.

5.4 Implementation of the tracking algorithm

The algorithm implementation to track an object using stencils is described here. Due to the constraints of the digital domain, only integer pairs are considered. So, instead of computing S_U, only $\mathbb{S}_U := S_U \cap \mathbb{N}^2$ is evaluated.

First, the pre-computing stage is described, and since it is conducted off-line, time is not an issue as long as computation times are acceptable. Figure 5.2 presents a flow chart of this stage. The set O of feature point coordinates that describe the shape is first extracted from an image that shows the object to be tracked. A Canny edge detector was used but any other feature detectors could also have been used. A trade-off between the number of features resulting from the filtering stage and the time it takes to extract these features has to be found when selecting and tuning the feature extractor. The same feature detector is used to filter images during live tracking and thus it has to be fast. Moreover, it is generally faster to identify an object if a few reliable features characterise it. This is true for the presented algorithm. A list data structure can be used to store this set of features.

The transformation space is then chosen to characterise the set of possible transformations the shape can undergo between 2 frames. According to the precision required to locate the shape, the transformation space is divided into a number of subsets such that the maximum distance between two transformations in a subset

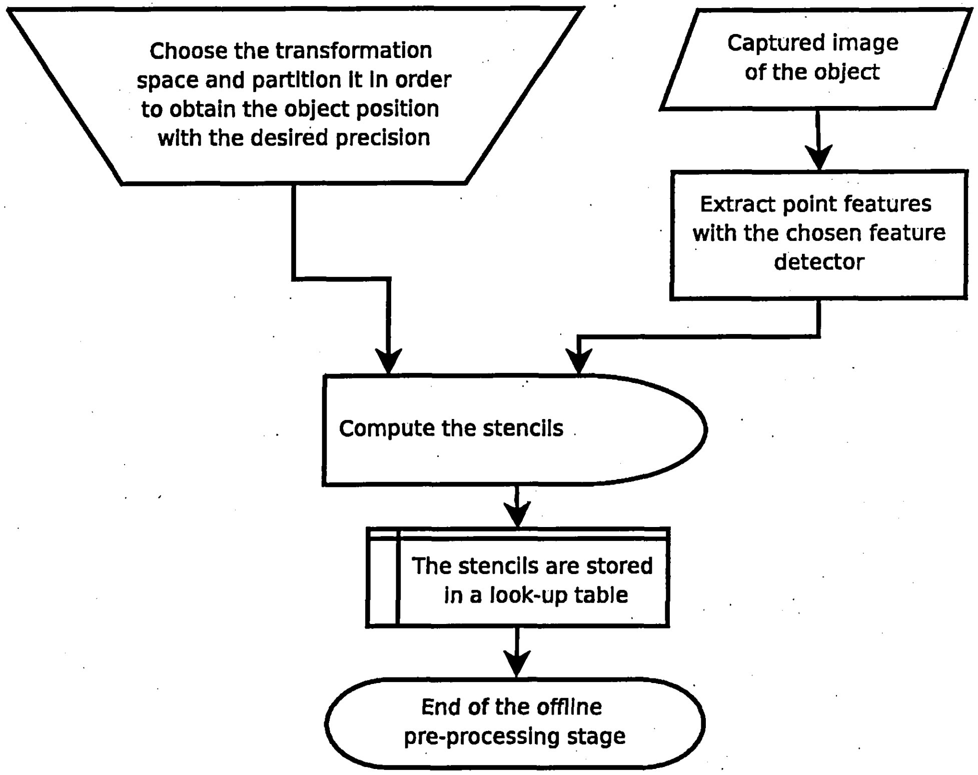

Choose the transformation space and partition it in order to obtain the object position with the desired precision
Captured image of the object
Extract point features with the chosen feature detector
Compute the stencils
The stencils are stored in a look-up table
End of the offline pre-processing stage

Figure 5.2: Pre-processing stage

does not exceed the precision needed. To implement this a multidimensional array, where each dimension corresponds to a transformation dimension, can be used. Each element corresponds to a sub-transformation set.

Then, the most computationally expensive stage of the pre-tracking stage is performed, the stencils are computed and a 2-D array containing list of references can be used to store them. These references point to the multidimensional array of sub-transformations. To obtain this 2-D array, the list of feature points is used and these feature points are transformed according to the set of sub-transformations that an element of the multi-array represents. For each coordinate obtained, a reference to the sub-transformation element is added to the corresponding elements of the 2-D array. Once this stage has been performed the 2-D array contains all the information necessary to identify a stencil. This 2-D array can be considered as the one-to-multiple element mapping that associates to each point $p \in \mathbb{N}^2$ the set of transformation sets of M that transforms a feature point of O to the point p:

$$\begin{aligned} \text{array}: \mathbb{N}^2 &\rightarrow \mathcal{P}(M) \\ p &\mapsto \{U : \mathbb{S}_U \cap p \neq \varnothing\} \end{aligned} \tag{5.7}$$

Indeed, for each element of the 2-D array that can be considered as a 2-D coordinate, a list of the references of the sub-transformations that transform a point of O to this coordinate is available. Given this 2-D array and a reference of the sub-transformation, the coordinates of the points of the corresponding stencil can be extracted by examining the 2-D array. This is the representation of $(\mathbb{S}_U)_{U \in M}$.

Figure 5.3 presents a flow chart of the tracking stage, it is performed as follows. The initial position of the object has to be determined. This can be done manually or using a detection algorithm. The detection algorithm is crucial in practice since, due to occlusion for instance, the tracking of the object can fail. The detection algorithm will then serve to re-initialise the tracking. Since detection algorithms are usually slower, because they are searching for the object in the whole image, it is not usually possible to use them directly for tracking. However, this distinction has become increasingly blurred over time due to the advancement of detection algorithms that are able to track objects under certain conditions.

The feature points, P, of the following image are extracted and their coordinates are expressed relative to the previous location of the shape. In practice, only feature points in the surrounding of the previous location of the object are needed. The coordinates are then used to look up the 2-D array of references. For each feature point, counters, c_U, where $U \in M$, corresponding to the references listed by the element of the 2-D array, are incremented. This is equivalent to voting for each of the stencils covering a feature point. At the end of this process

$$c_U = \sum_{p \in P} \begin{cases} 1 & \text{if } U \in \text{array}(p) \\ 0 & \text{else} \end{cases}$$

The reference that has the highest count is then considered to correspond to the subset of transformations that contains the transformations the shape has undergone. A transformation from this subset is taken, for instance, one that minimises a distance with the other transformations of the subset, and, the new estimated position of the shape is evaluated by combining this transformation with the previous estimated position of the shape.

For more details, the reader is invited to refer to appendix E where a minimal C++ implementation for tracking objects that translate in the 2-D plane is provided.

For the EU FP6 MiCRoN project a more complex implementation was developed.

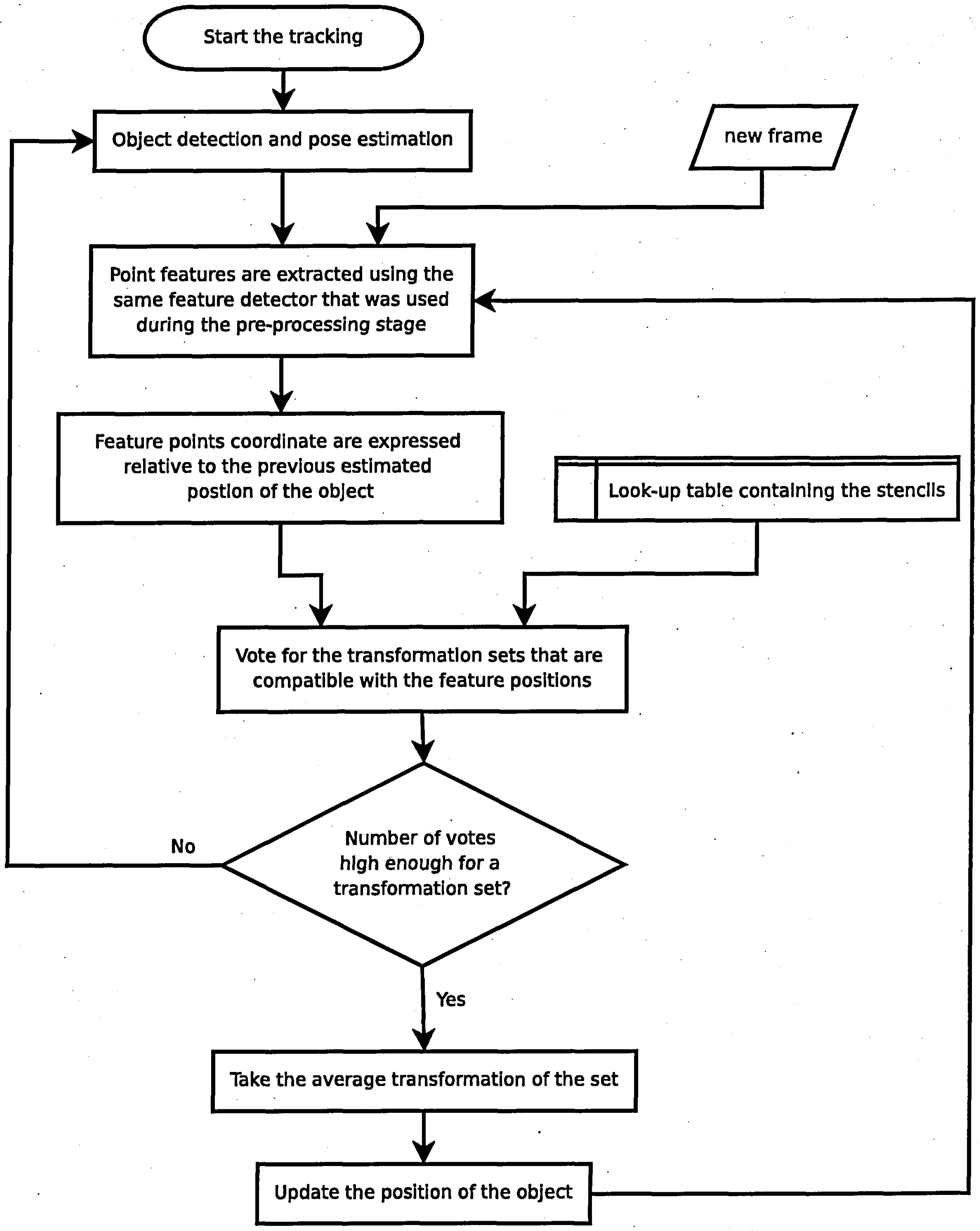

Start the tracking
Object detection and pose estimation
new frame
Point features are extracted using the same feature detector that was used during the pre-processing stage
Feature points coordinate are expressed relative to the previous estimated postion of the object
Look-up table containing the stencils
Vote for the transformation sets that are compatible with the feature positions
Number of votes high enough for a transformation set?
No
Yes
Take the average transformation of the set
Update the position of the object

Figure 5.3: Tracking stage

To be able to track objects depth-wise a stack of images was used. Indeed, because of the narrow depth of field of microscopes when objects are moving relative to the microscope lens their appearance changes. More details on how to take advantage of the focus effect for microscope images is provided in section 6.1 page 100. Rotations were also taken into account and it is shown how the algorithm can be parallelised. An analysis of the time and space complexity of the algorithm is also discussed.

The implementation was tested and combined with a detection stage as part of the MiCRoN project. The final software is available on the web and can be found in the MMVL wiki page[1]. Other implementations are available in the Mimas library [97].

5.5 Stencil reduction

When profiling the tracking algorithm, most of the time is spent incrementing votes. This suggests that reducing the number of stencils covering each feature point would improve the speed of the algorithm. Moreover, most of the memory space is used to store the look-up table containing the stencils. Reducing the size of each stencil will reduce the memory footprint. Note that the algorithm speed and memory usage was not an issue for our tracking application. However, there are three reasons to do this:

- In embedded systems, memory and processing power are limited.
- The number of degrees of freedom of the tracking algorithm can be increased.
- Tracking is often only one component of a larger system. So if fewer resources are used by this stage then more can be used by other stages, either by lower level, or higher level algorithms.

The 2-D array structure that has been used to store the stencils can be used to achieve this aim. In the previous chapter we mentioned reducing shapes, however this idea emerged chronologically after we tackled this issue and with hindsight, the issue would have been tackled slightly differently. In other words, instead of directly reducing the template of the object to a characteristic set of points and then create stencils from it, the stencils were first generated and then reduced. The

[1]http://vision.eng.shu.ac.uk/mmvlwiki, October 2007

reduction has to be done in such a way that a set of feature points that correspond approximately to the searched shape would still trigger the same stencil, even after the area reduction of the stencils.

Considering the 2-D array mapping that associates a list of references to a features point p, the time to increase all references is proportional to the size of the list, *i.e.* $|\text{array}(p)| := \text{card}(\text{f}(p))$. However, when the list of references is long, not only does the point not contribute much information, since any of the references might contain the transformation that is being looked for, but it also takes more time to account for these features than a feature that would contribute more relevant information.

This led to the idea of decimating the points of the stencils that are overlapped by many other stencils. In order to retain robustness to occlusions that may happen on localised area, *i.e.* on one part of an object, the decimation should be done across the whole stencil. Moreover, the area of a stencil should remain large enough to receive a significantly larger amount of votes when it covers the shape than the number of votes received by other stencils. To implement this we proposed algorithm 2.

Algorithm 2: Stencil decimation

```
foreach stencil $S$ do
    while the stencil area is too large do
        foreach point $p$ of $S$ do
            calculate $|\text{array}(p)|$
            evaluate $\sum_{p \in S} |\text{array}(p)|$
            randomly eliminate a point according to its weight $g(\frac{|\text{array}(p)|}{\sum_{p \in S} |\text{array}(p)|})$
Where $g$ is a monotonically increasing function.
```

This algorithm was tested in various ways using the identity function for g. The results are presented and discussed in section 6.2 page 112. We mentioned the g function because it is unlikely that the identity function provides the best results. The random elimination of a point according to its weight was directly inspired from the propagation stage of particle filters. The higher the weight of a point, the more likely it is to be removed. As tests show in section 6.2, the method works fine but much more investigation is needed to understand why and to explore how to improve it. The previous chapter had presented such an attempt.

The presented algorithm is just one of many ways to decimate the stencils. Although no direct (algebraic) link can be established with the margin of error[2], as has been defined in this chapter, it seems logical to think that the more a stencil is overlapping another stencil the more likely it might be responsible for an incorrect estimation of the shape state. Hence, a policy that favours the reduction of the area of a stencil in its area that are overlapped by its most overlapping stencils[3] may decimate the stencil in a way that does not alter the robustness significantly.

The random picking of the stencil points help remove points from the whole area resulting in a possibility for points having a high weight to remain. The tests that have been conducted show that by using this approach, the remaining points are spreaded over the whole stencil area. We propose another scheme to maintain a uniform point repartition over the stencils in order to ensure that the stencil is robust to occlusion: a map of the stencil could be used to decrease the probability of a stencil point to be selected when a previously discarded point lies in its neighbourhood. In other words, the weight associated with each point of a stencil could be altered to decrease the probability of a point to be removed if a neighbouring point was previously removed.

Lastly, the minimum area size of a stencil has to be carefully considered. It should be as small as possible to optimise speed and memory but large enough to resist features that are not generated by the tracked shape. The next section provides some guidance on parameter tuning.

5.6 Summary

It has been demonstrated that in ideal conditions the stencil estimator can be used to uniquely determine the location of a shape when the transformation space consists of translations. In general this property does not hold when a transformation space has a higher number of dimensions, for instance, when scale changes or rotations are considered, however, in practice, and especially if the set of points of the shape are sparse, the stencil estimator yields almost always the expected result. This will

[2]Because the transformation that defines the margin of error might not be covered or only partially covered by the stencil that most covers the stencil under consideration.

[3]Determining the most overlapping stencils is easy to implement using the tracking algorithm: the set of points of the stencil can be fed into the tracking algorithm. The number of votes obtained by a transformation space correspond to the size of the overlapped area

be visible from the experiments presented in section 6.2, page 112

A few definitions were proposed to characterise the robustness of the stencil estimator. Then the implementation of the stencil estimator using the bounded Hough transform, that is sometimes referred to as the stencilled Hough transform, was discussed. Finally an algorithm to decimate stencils in order to increase performances without altering significantly the robustness of the tracking is proposed and discussed. Testing of the stencilled Hough transform algorithm and the decimation algorithm on synthetic data are provided in section 6.2 page 112.

Chapter 6

Experiments

6.1 Tracking of Microscopic Objects

6.1.1 Context and experimental setting Microscope images have a very narrow depth of field. Consequently part of the object can be in focus while the rest is out of focus as shown in figure 6.1. In these images, 1 pixel translates to approximately 1 μm and the field of view is about 1 mm^2. The stencilled Hough transform has been modified to make use of the fact that the global appearance of the object changes with its distance to the camera. Figure 6.2 shows a diagram of the set-up used for the experiments.

In order to estimate the depth position of the object, a stack of images of the object, taken at different distances from the lens of the microscope, is used. The model of the object consists of the features extracted from this stack of images. Figure 6.3 illustrates the model of the micro-gripper that can be seen in figure 6.1. The vertical distance between each pair of images is approximately 10 μm. Thus a better resolution is achieved for the horizontal translations than for the translation along the axis of the microscope. The top left corner image is at the bottom of the stack and the stack is sorted left to right and top to bottom. Since edges are blurred when the gripper goes out of focus, there are fewer features at the beginning and the end of the stack. The maximum number of features is obtained when most parts of the gripper are in focus. So as to obtain these features, as speed is critical during tracking, a simple feature detector that is fast was used.

Let L be the look-up table of the stencils that are created by moving the model image according to all sub-transformations. This look-up table is a 2-D array and,

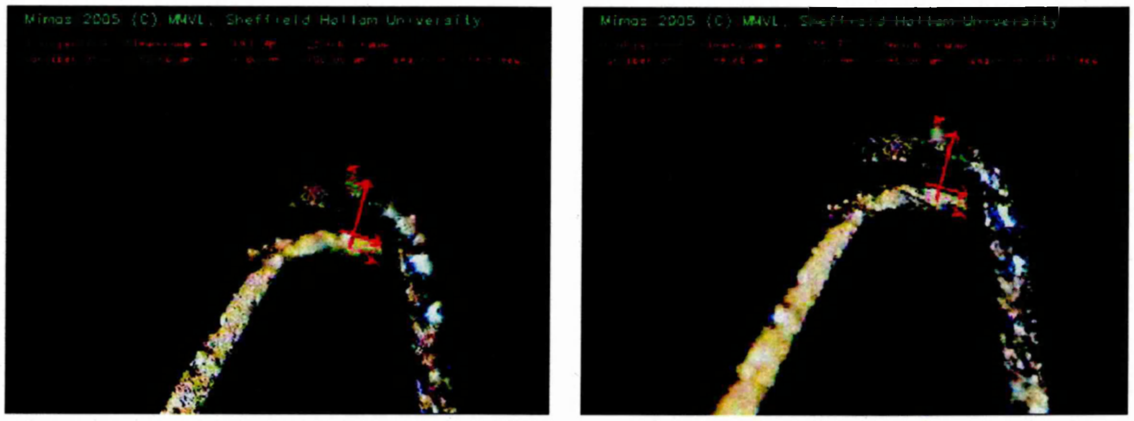

Mimas 2005 (C) MMVL, Sheffield Hallam University
Mimas 2005 (C) MMVL, Sheffield Hallam University

Figure 6.1: Images from the gripper tracking sequence. The grippers in the left and right images are at different depths.

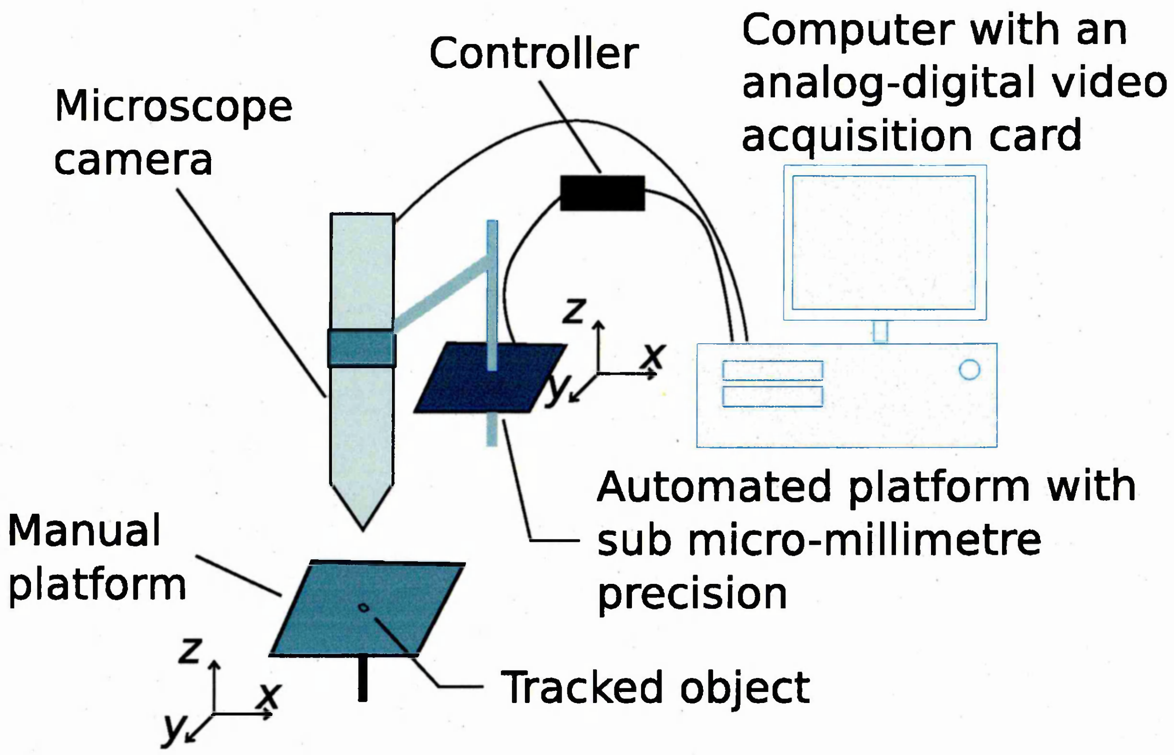

Microscope camera
Controller
Computer with an analog-digital video acquisition card
z
x
y
Automated platform with sub micro-millimetre precision
Manual platform
z
x
y
Tracked object

Figure 6.2: Diagram of the set-up used for the experiments. A photo of the set-up can be found figure 1.4, page 6.

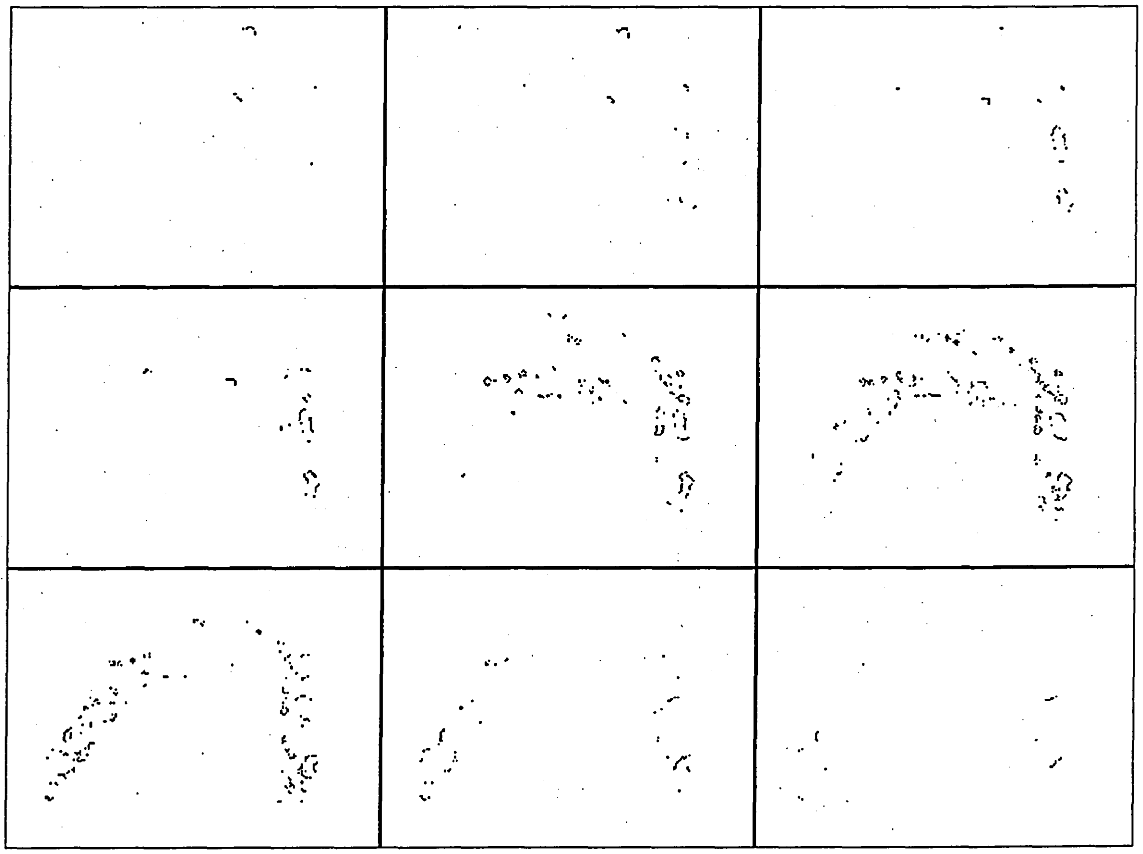


Figure 6.3: A subset of the stack of images that serves as the gripper model

in the gripper case, its size was about 20 % larger than the model image. This value depends on the selected transformation space which depends on how the object motion can be bounded. Conceptually, during the tracking phase, the look-up table is positioned over the image at the previous position of the tracked object. The stencil that encompasses the most features, relative to its size, is considered to indicate the displacement that the object has undergone. A reference point was taken at the centre of the model images. Let $f = (x, y) \in \mathbb{N}^2$ be the position of an edge feature relatively to this reference point. In this case, the transformation space is a set of $\mathbb{R}^3 \times \mathbb{N}$ and we denote $S = \{[T_{x1}, T_{x2}], [T_{y1}, T_{y2}], [\theta_1, \theta_2], n\}$ a sub-set of this space. Rotations are centred about the reference point. By selecting the reference point at the centre of the images of the stack, the displacement of the features due to the rotational of the stack is reduced component and results in a more compact look-up table.

6.1.2 Algorithm

Pre-processing Stage

The following procedure can be used to obtain the stencil associated with a subset of transformations S:

1. Select the image i of the stack that corresponds to the depth component of S.

2. Consider the rotation part of the subset of transformations; each feature f is transformed in a set of points that forms an arc centred about the reference point, with angle $\theta_2 - \theta_1$ where the middle point of the arc is f rotated around the reference point by $\frac{\theta_1+\theta_2}{2}$. To determine the pixel points, or digital points, that correspond to this description, $[\theta_1, \theta_2]$ is sampled every $\delta\theta$. Tests show that for the tracking of the gripper 0.5° is sufficiently small. However, this value depends on the model size, which is expressed in pixels. Thus, a set of digital points P_1 is obtained. If a digital point is selected multiple times it should be considered only once.

3. Consider now the translation part $[T_{x1}, T_{x2}]$. For translations, the natural distance is expressed in pixels. Microscope calibration can be used to determine the exact corresponding real-world distance. As the transformation

space is divided, the bounds of the segment T_{x_1} and T_{x_2} may not be integers. Different policies to determine a range with integer bounds can then be applied. We first chose to consider the smaller range with integer bounds that include $[T_{x_1}, T_{x_2}]$. However, when the object moves in the middle of two ranges, two stencils are likely to receive a high count of features. Another policy would be to take the closest integer for each bound. Once a choice has been made, the set of points P_2 obtained by the translation is calculated. $P_2 = \{(x + t_x, y) : t_x \in \left(\text{pl}([T_{x_1}, T_{x_2}]) \cap \mathbb{N}\right) \mathit{and}\, (x, y) \in P_1\}$ where pl is a function that describes the chosen policy.

4. Proceed in the same way to obtain P_3, the set of digital positions obtained by translating the set P_2 along the y-axis.

5. P_3, which corresponds to the positions of the model feature if the model was moved by the set of transformations of S, is used to fill L by adding a reference to S corresponding to each point of P_3.

Figure 6.4 illustrates some of the stencils obtained. Although it may not seem apparent, stencils with the same shape are slightly translated or rotated relatively to each other. Different shapes in this case correspond to stencils coming from different images of the stack.

Tracking Stage

The tracking stage is now described. Let $\hat{p}_t := (x, y, z, \theta)$ be the estimated position of the object at frame t through four degrees of freedom. The tracking is performed as follows:

1. The region of interest (RoI) of the image is selected. It is centred on (x, y) and has the size of L and orientation θ.

2. Features are extracted using the same edge detector used to extract the model features and their position is expressed relatively to the centre of the RoI with a frame rotated by θ.

3. For each feature, we examine L at the corresponding position of the feature and increment the corresponding stencils by 1.

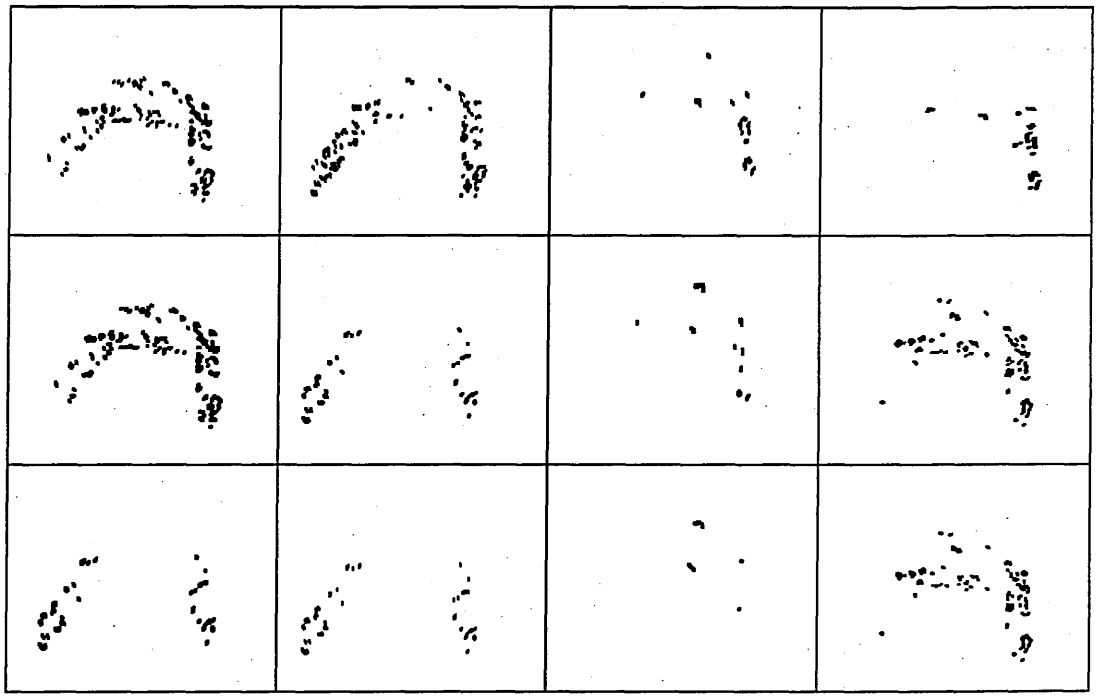


Figure 6.4: A sample of some of the stencils produced

4. The stencil with the greatest hit number (votes) is considered to be the one associated with the sub-transformation S that contains the movement of the object.

5. $\hat{p}_{t+1} = (x + \frac{T_{x1}+T_{x2}}{2}, y + \frac{T_{y1}+T_{y2}}{2}, n, \theta + \frac{\theta_1+\theta_2}{2})$

A working system combined with a recognition algorithm [37] has been successfully implemented. The algorithm implementation, which relies on the open source Mimas vision toolkit framework [97], as well as some test data are available on the Internet [98]. It serves as a good starting point to further test the algorithm.

6.1.3 Complexity analysis Experimental data that demonstrates how parameter variations affect the speed and space complexity of the algorithm is hereby provided. In order to carry out this comparison we tracked the micro-gripper shown previously on a video sequence of 470 frames.

The feature extractor was tuned to obtain a successful tracking in more than 90 percent of the 470 images. We then obtained a model that had 10 focus planes having respectively from top to bottom containing 17, 33, 51, 75, 206, 324, 258, 71, 26, 10 features. Figure 6.3 shows the model for the gripper. The last image has not been shown since it does not hold enough features and is not relevant for the tracking. Figures 6.6, 6.5 and 6.8 summarise the results of the tracking experiments carried out with 4 different sets of parameters.

Let us first examine the pre-processing stage. If N is the number of model features and O the number of possible discrete transformations, by construction of L, a maximum of $N \times O$ feature locations have to be determined and stored. Thus, $\mathcal{O}(N \times O)$ represents both the maximum space complexity of the algorithm and the speed complexity of the pre-processing stage. Notice that O changes with the power of the transformation dimensions *i.e.* doubling the size of the transformation space multiply by approximately 2^d the memory usage and the pre-processing time, d being the number of dimensions. However, if we look at figure 6.6, the outcome is unexpected. The complexity analysis needs to take into account another important factor: how the state space is divided. The time complexity for the pre-processing stage is not affected by this factor since the transformed models have to be evaluated for every digital transformation. However, the space complexity depends on how the transformation space is divided. To understand this consider a few transformed models, a certain number of their points will be the same, if these transformations belong to the same subset of transformations of the state space all these points will refer to only one subset of transformations, but if the transformations belong to different subset some of the points will refer to multiple subsets therefore increasing the size of the memory space used. When the points are not sparse, which is the case with the feature detector we are using, this phenomena is not negligible.

Returning to figure 6.6, the parenthesised values allow easy comparison of timings and memory usage between the different tests. Reference value 1 has been taken for the first experiment on the top left corner of the table. First of all notice that it is not possible to double the depth precision as we have chosen a value close to the limit of the depth resolution. If we take more images of the object they will not be different enough to be discriminated by the algorithm. However, it would be possible to increase the resolution on the depth axis by having a camera with a narrower depth of view which would also bring a narrower operating field to simultaneously see different objects.

From the above analysis precision should not affect the memory usage. However, figure 6.6 shows that the memory usage has increased significantly (2.77 times more). This is partly due to our policy to include pixels that intersects a transformed feature

How dimension are divided:
translation along x, along y, depth, rotation
Dimension sizes:
pixel, pixel, number of images, degree
Preprocessing time
Number of elements stored in L
Asymptotic behaviour of the tracking

Figure 6.5: Key for figure 6.6

even if the intersection is very small. The effect is not negligible because our stencils are "thin".

It can be noticed, when comparing the top-left and the bottom-left parts of the array, that increasing the transformation space by 8 results in an increase of 6.19 in the space complexity, this is because rotational transformation unlike translations do not increase proportionally the stencil area.

The tracking stage is now considered. When looking for the object, a region of interest, having the same size of L and which holds P image features, is considered. Considering that on average each element of L refers to a constant number of stencils then the tracking complexity should be approximately proportional to the number of features present in the region of interest, so the average time complexity should be $\mathcal{O}(P)$. Figure 6.8 confirms that this hypothesis is a good approximation, the linearity between the number of image features and the time to process an image appears clearly. To further demonstrate this relationship we plotted, figure 6.9, the number of votes against the number of features of a tracked image. Notice that on figure 6.9 the trend line forks for small numbers of image features. This links with the similar pattern that is visible on the bottom right graph of figure 6.8.

Figure 6.6 and 6.8 show that the speed varies with the precision and the size of the search area in a non trivial way. The variation depends, amongst others, on the feature density and the shape of the object model. The trade-off between speed and precision and between the size of the search area boundary and the speed of the tracker is clear. The complexity analysis provides an idea of the trend behaviour of the speed of the tracker, so as to be able to forecast and guarantee the real speed of the algorithm requires to test the implementation.

	Precision	
Tracked area	4, 4, 10, 2 (1) 20, 20, 10, 4 (1) 12.13 s (1) 561 648 (1) $2.30 \times 10^{-2}x + 1.2$ (1)	9, 9, 10, 3 (7.59) 20, 20, 10, 4 (1) 21.34 s (1.76) 1 556 776 (2.77) $7.1 \times 10^{-2}x + 6.79$ (3.1)
	9, 9, 10, 3 (7.59) 40, 40, 10, 8 (8) 68.29 s (5.63) 3 477 125 (6.19) $9.77 \times 10^{-2}x + 0.87$ (4.25)	18, 18, 10, 6 (60.75) 40, 40, 10, 8 (8) 166.93 s (13.71) 11 882 872 (21.15) $28.2 \times 10^{-2}x + 42$ (12.26)

Figure 6.6: Tracking using 4 sets of parameters, see figure 6.5 for keys and figure 6.1.3 for the corresponding tracking behaviour

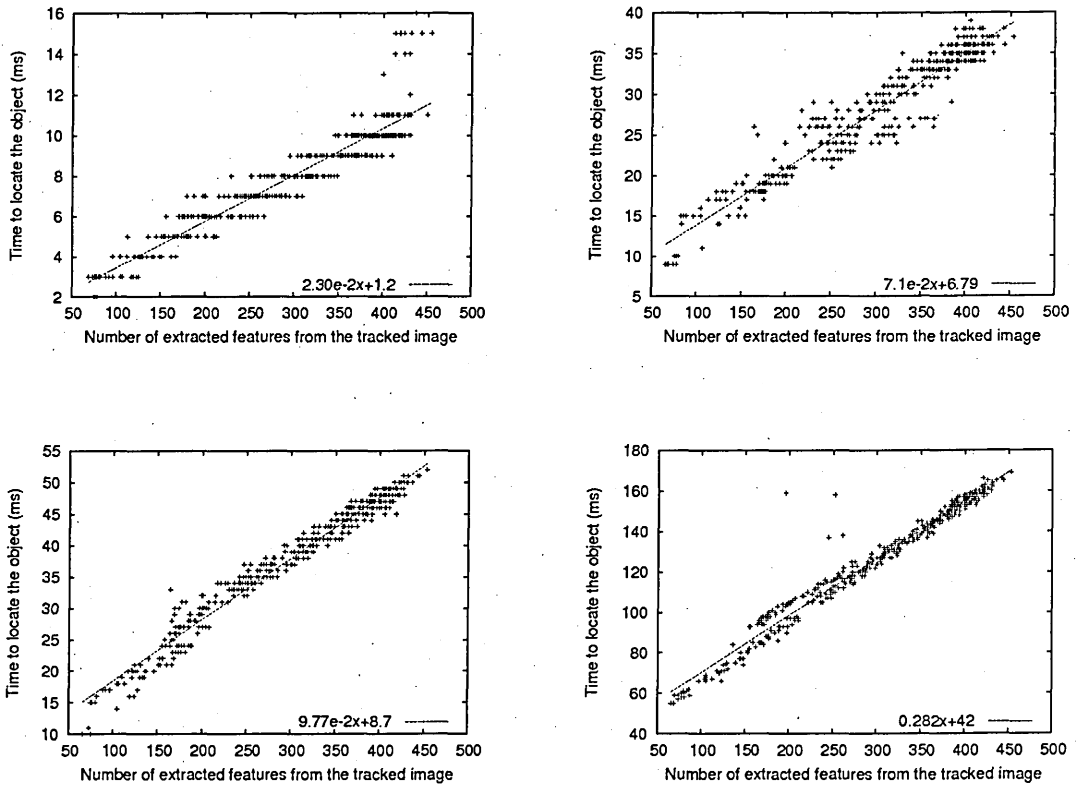


Figure 6.7: Speed comparison of the tracking of the gripper using different parameters

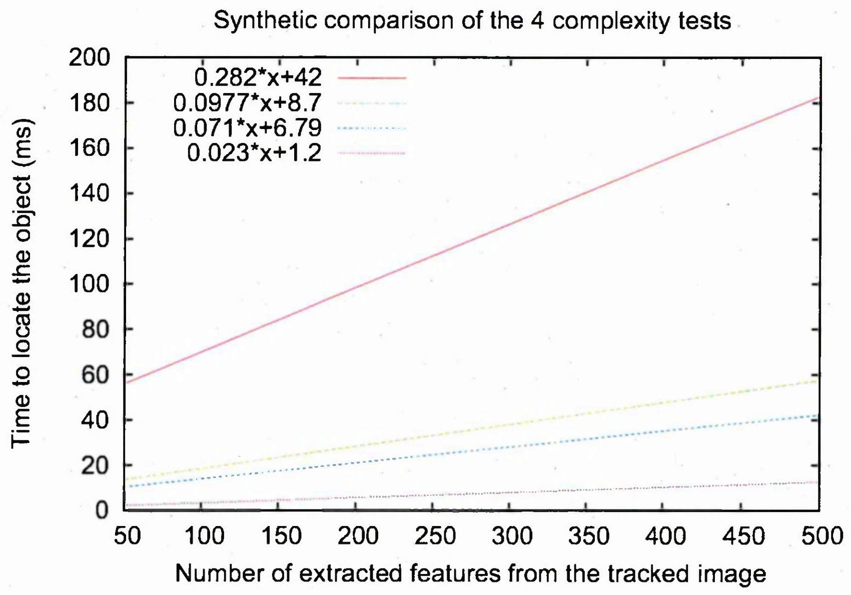


Figure 6.8: Visual comparison of the asymptotic tracking behaviour

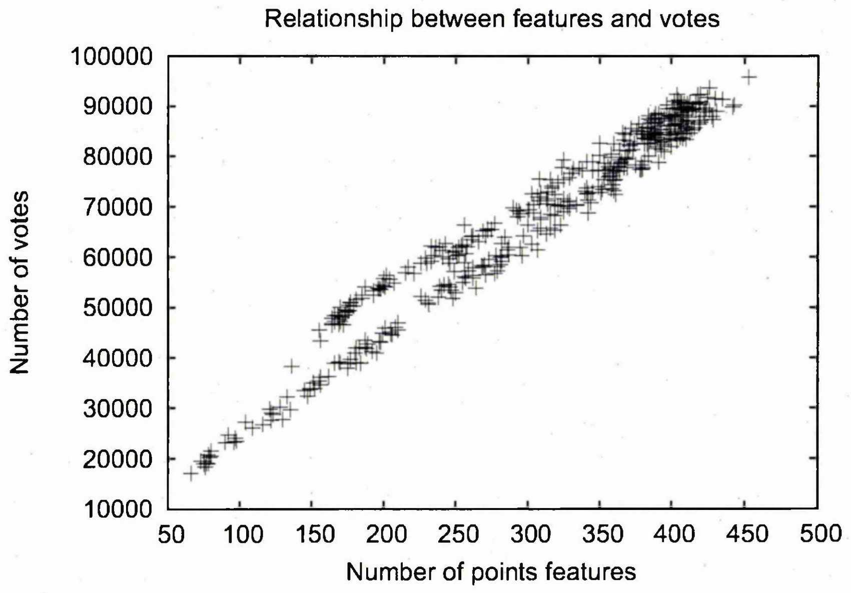


Figure 6.9: Number of stencil increment versus number of image feature

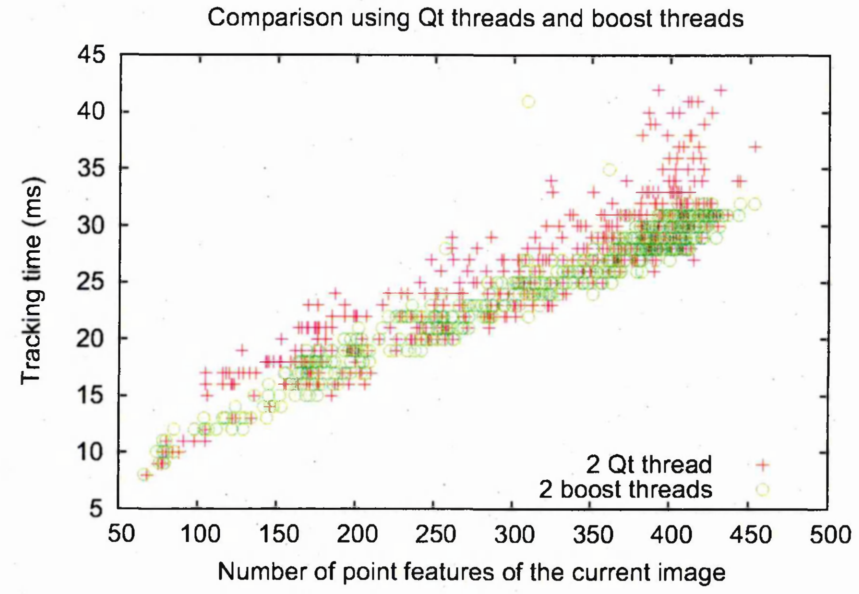


Figure 6.10: Comparison of Qt threads and boost threads, both implementation uses Qt mutexes

6.1.4 Parallelisation The algorithm can be easily parallelised as follows: for each thread/processor we associate an array L in which a subset of the stencils is stored. The tracking can thus been carried out simultaneously by different threads/processors.

Experiments were done with different implementations using threads from the Qt library, Boost library and different synchronisation primitives (mutex, condition and barrier).

In our test sequences threads from the Boost library were marginally faster and presented more consistent timings (perhaps due to the display thread of the Qt library that was used and that could have interupted more often the Qt threads) as shown in figure 6.10. It was also found that implementations which launch new threads for each new image will track slower. We were unable to find a reason why, since launching a thread should be extremely fast, we suspect that this is due to cache misses on the CPU. Figure 6.11 compares a few implementations of the parallelisation. The best implementation obtained was by using permanent Boost threads synchronised with barrier primitives.

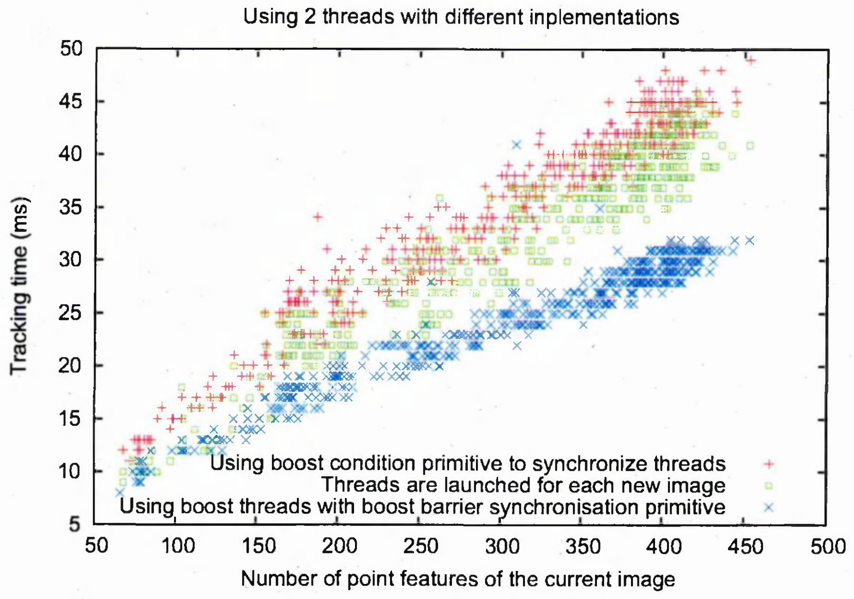


Figure 6.11: Comparison of the parallelisation on 2 processors using different implementations

The best implementation was then used to compare the tracking speed using one and two threads. The results are shown in figure 6.12. The asymptotic speedup is $\frac{0.089}{0.056} = 1.59$ which corresponds to an asymptotic parallelisation efficiency of $\frac{1.58}{2} =$ 79%. Notice the few outliers that appear on the different figures which are probably due to time slicing.

6.1.5 Summary In this section it was shown how the stencilled Hough transform can be used to track rigid objects under a microscope in real-time (12 fps here). The change in appearance of the object has been taken advantage of to track the object depth-wise.

One pitfall is a lack of a recovery mechanism when the tracker fails, for instance if the gripper goes out of focus. To solve this, the tracker has to be coupled with a robust recognition algorithm that also serves as the initialisation process for the tracker.

The technique was adapted to track microscopic objects with 4 degrees of freedom in images with limited depth of field. The usage of colour cues, edge orientation or

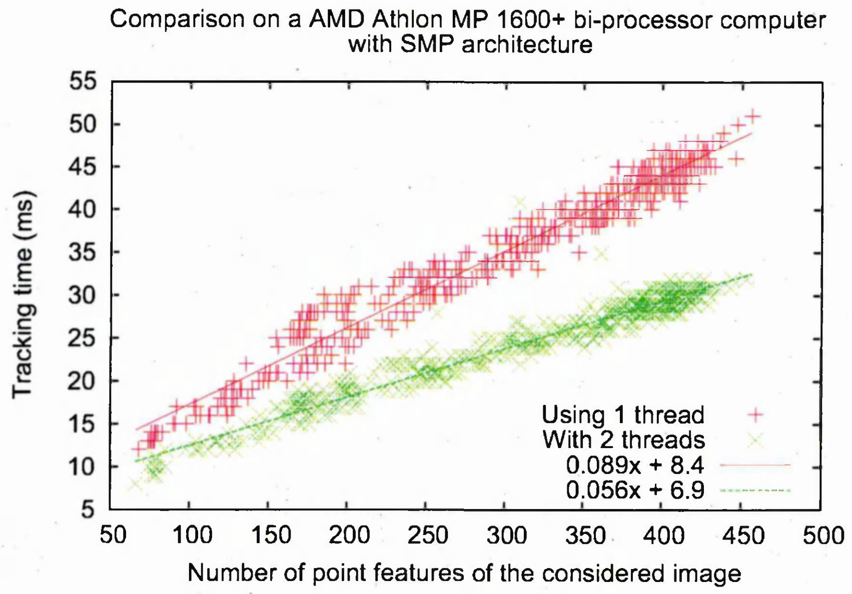


Figure 6.12: Speed comparison, one thread versus two threads

different kind of features are a few of the possibilities to further improve the speed and robustness of the algorithm.

Finally, we have shown that parallelisation significantly increases the speed of tracking. An implementation in C++ as well as test data is available on the web under the title "MiCRoN vision".

6.2 Testing of the stencilled Hough transform using synthetic data

6.2.1 Experiments and results In order to test the robustness of the stencil tracking algorithm, in a controlled manner, artificial images were generated. An arbitrary set of points was chosen and translated across an empty image. The displacement of the shape from one image to another was randomly selected in the range allowed by a chosen transformation space. Also, the movement of the shape was constrained to a certain distance out of the bound of the image. By using an arbitrary set of points, results were obtained independently of a given filtering method.

A pixel can be in two different states, a "feature point" state or a "no feature point" state. To evaluate how the stencil reduction method copes with disturbances, salt and pepper noise was added to the image sequences. Salt and pepper noise provides an extreme test case not present in real sequences, which enables the testing of the algorithm for robustness. The salt and pepper noise was uniformly distributed and expressed by its presence likelihood. It was generated as follows: for each pixel a random number between 0 and 1 is generated and according to the desired rate of error the pixel state is set to its other state. The loss of information is maximum when the likelihood is .5. Indeed, whatever the original state of a pixel, both states are equally likely to occur after noise of that level has been added.

The rate of false positives and false negatives were not considered separately. In practice false negatives, due to the feature detector for instance, tend to happen less often than false positives, due to the background. However, such a detailed analysis would have complicated the generation and the interpretation of the tests. Although this kind of noise has little relationship with what happens with real images, it gives insight in how the algorithm behaves when severe disturbances are present in the image. As a simple example, when illumination changes due to the change of direction of its source, movement of the camera, shadow or movement of the object, it is not unlikely that a few features will shift slightly relative to each other. This generates false positives as well as false negatives, it is however complex to model and it is only one phenomena out of a multitude of others. Therefore a realistic modelling of disturbances is highly complex and for our purposes not within the bounds of this thesis.

The results obtained by comparing the ground truth of the shape position from the synthetic images we have generated with the results of the tracking algorithm are now described.

To carry out our tests, two sets of image sequences were generated. Figure 6.13 shows the template models used to generate the 2 sets of sequences. The same template were used to track the object. For visibility reasons, the hand shape size has been displayed 3 times larger than the watch shape. Note that although a hand is a deformable object the stencilled tracking is not appropriate, at least in its current form, to track non rigid objects, the hand shape is simply an arbitrary rigid shape and any other set of points would have been convenient for testing. The first data set was generated using the hand-drawn "hand" shape, containing 194

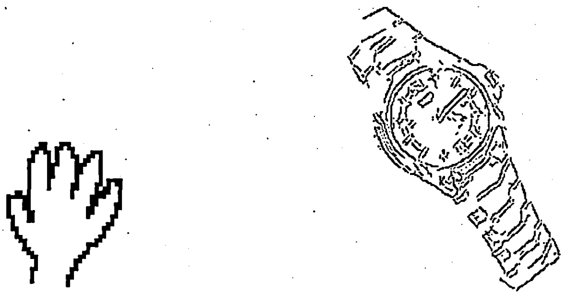


Figure 6.13: Templates of the tracked shapes.

features, translated over 500 images; from this sequence 26 other sequences were generated with different levels of salt and pepper noise. Figure 6.14 shows the first images of some of these sequences and helps visualise the level of noise present in each sequence. The second data set features the edge feature points extracted from an image showing a watch. The shape contains 2812 features. The image sequences consist of 800 images each. Again, sequences with different levels of noise were generated and figure 6.15 helps appreciate the content of noise in these images. For each template we generated 26 videos with different levels of noise and these videos were tracked with 11 different levels of stencil decimation. Therefore, for each template, the tracking was performed on 286 videos of 500 and 800 images each respectively.

When examining the figures in figure 6.14 the hand shape can be distinguished easily by the author's vision system until 28% of noise is present. With 34% and up to about 40% of noise, the shape can still be distinguished by the human brain. If the image sequence is viewed as a video, perhaps due to the shape movement, it is possible to guess more easily where the hand shape is at 40% of noise. For the watch shape shown in figure 6.15, maybe because the number of features is an order of magnitude higher, the shape position can be guessed more easily than the hand shape at 42% of noise. However, perhaps with a stretch of imagination, the silhouette of the shape appears with 46% of salt and pepper noise indicating the shape position. We will return to this point later, for now it suffices to say that the performances to locate an object using stencils appear to be similar to the human vision system.

Figures 6.16 and 6.18 sum up the tracking performances of the stencilled Hough transform, each point represent the number of erroneous locations made by the tracker for a given video sequence. The z-axis corresponds to the number of frames

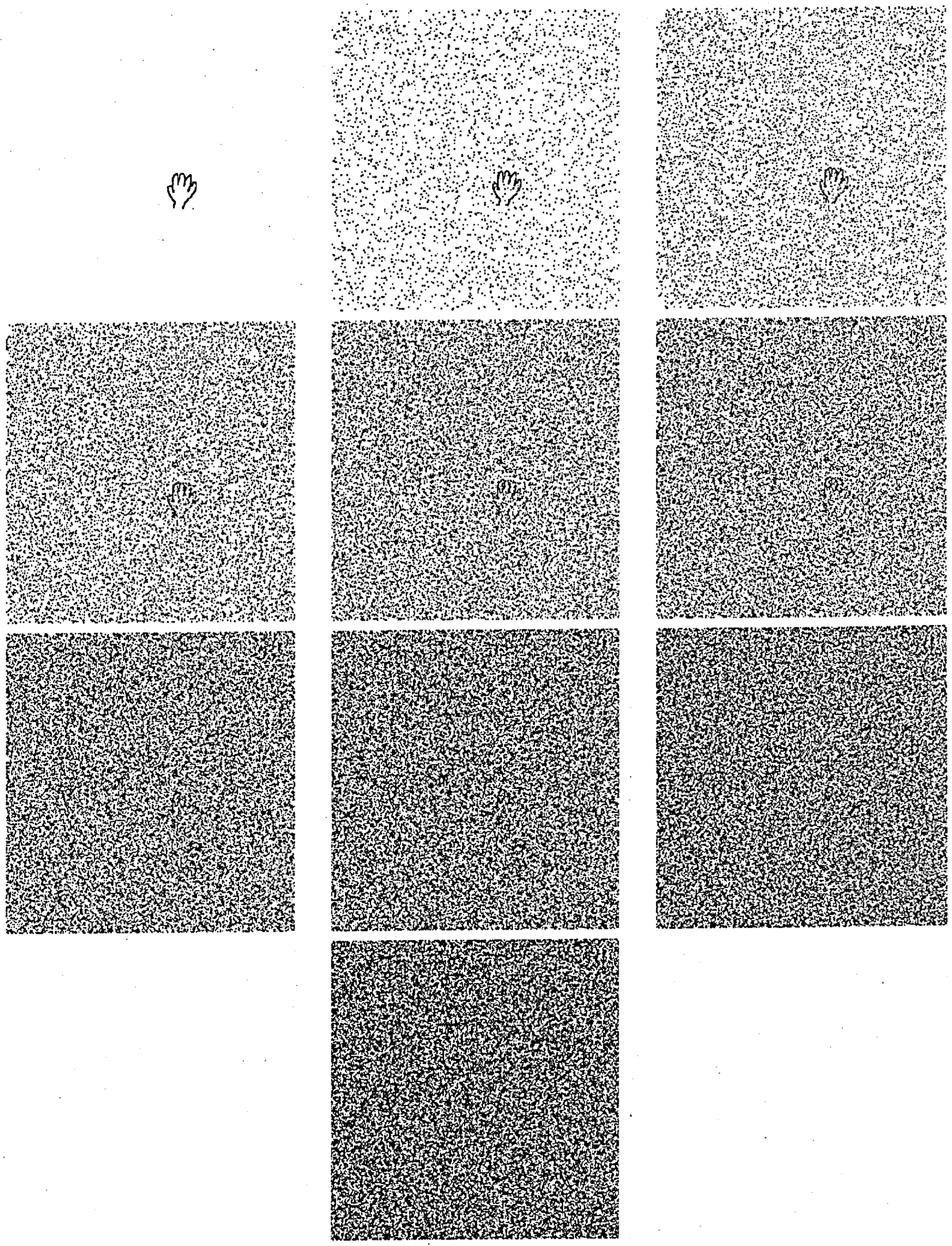


Figure 6.14: Appearance of the first image of the hand shape tracking sequence for different noise levels. In percentage of image noise: 0, 4, 10, 16, 22, 28, 34, 40, 46 and 50% respectively

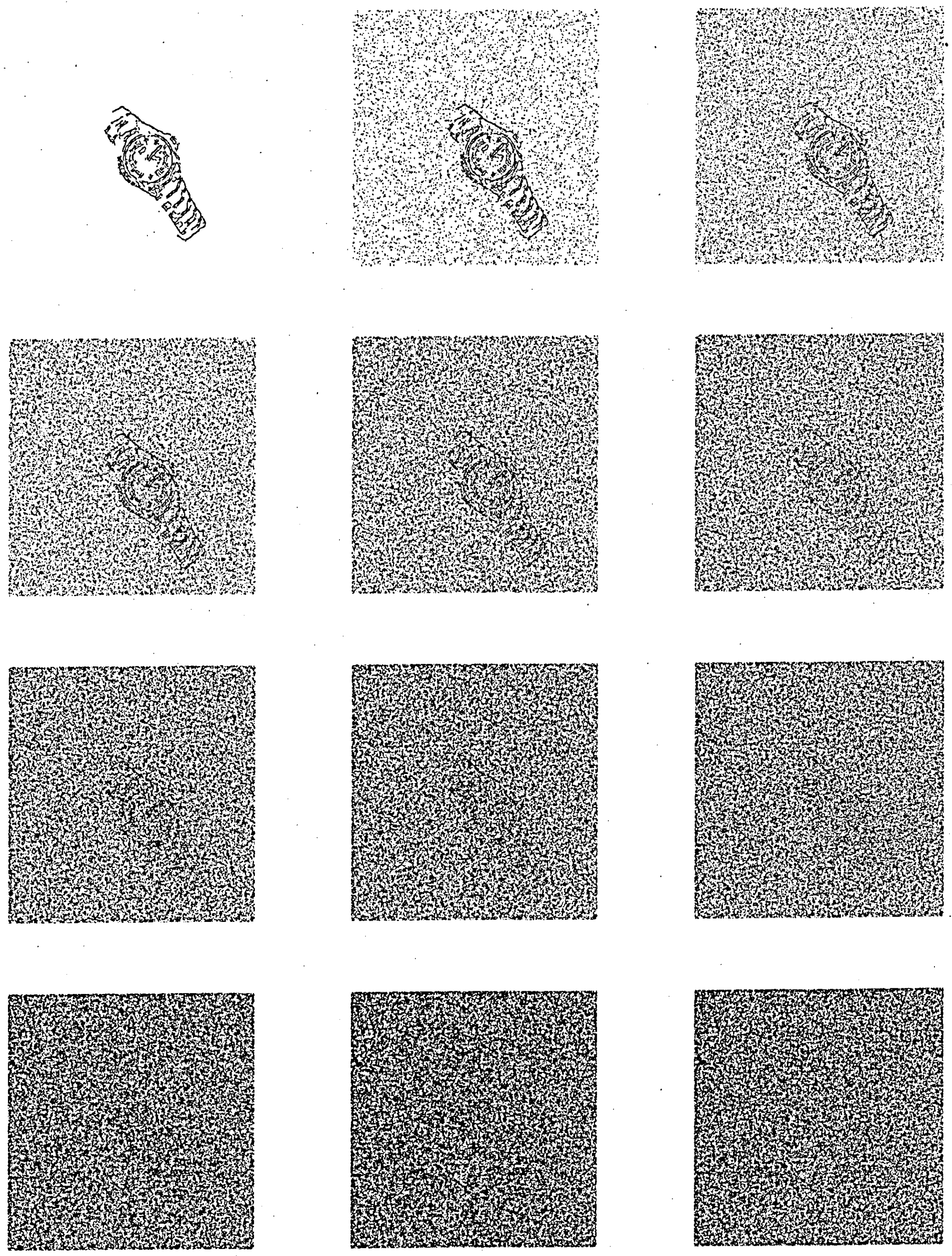


Figure 6.15: Appearance of the first image of the watch tracking sequence for different noise levels. In percentage of image noise: 0, 4, 10, 14, 20, 26, 30, 34, 38, 42, 46 and 50% respectively

where the tracking was incorrect, the x-axis corresponds to the noise of salt and pepper in the video sequence and the y-axis to the level of decimation of the stencils that was used to track the object in the sequence. Figures 6.17 and 6.19 represent the same data with level lines. By examining these figures it can be observed that, for the hand shape, the maximum capabilities of the stencil estimator, before any decimation of the stencils, to correctly track the shape is roughly 40% noise, *i.e.* 80% of the maximum possible level of salt and pepper noise. The outlier at ratio 0.8, 32% noise in figure 6.16 is due to the failure of the tracking algorithm at frame 348.

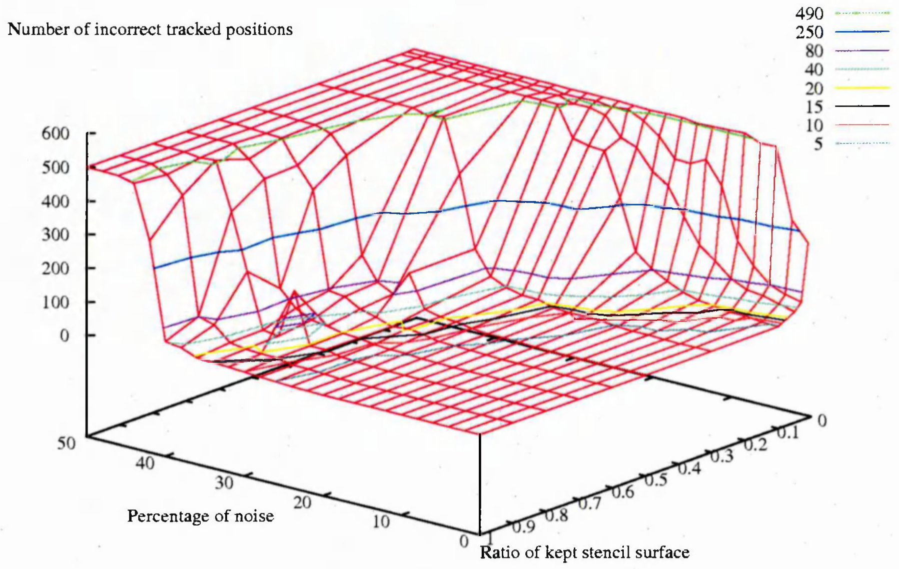


Figure 6.16: The z-axis corresponds to the number of images where the tracking result differs from the ground truth. The ratio of kept elements of the stencil is actually the ratio of the number of references listed in the 2-D array corresponding to the stencil elements kept. The colour lines outline error levels. This graph corresponds to the hand shape image sequences.

By examining figures 6.18 and 6.19 it can be seen that the tracking capability against noise for the watch shape is slightly higher at around 46%, that is 92% of the maximum possible level of noise. This is likely due to the much higher number of features resulting in a higher likelihood to reach a critical threshold of features to discriminate the shape from random noise. Arguably, and judging from figures 6.14 and 6.15, a level of salt and pepper noise of around 20% is well above the maximum

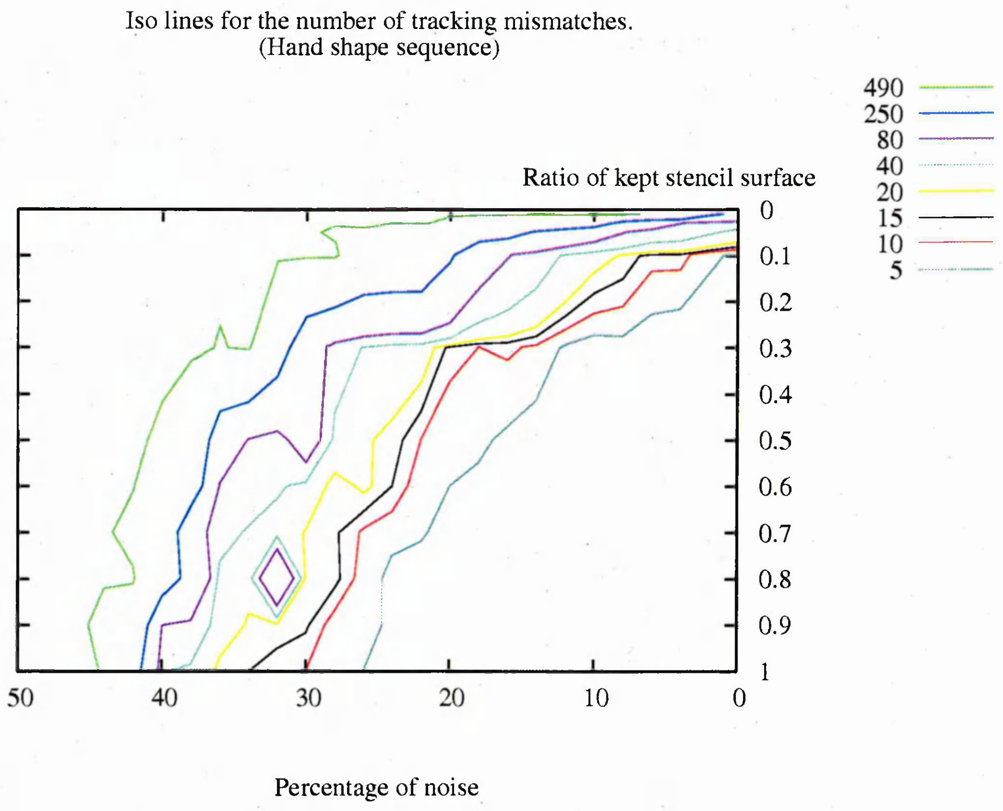


Figure 6.17: The same graph as in figure 6.16 viewed from the top and with just the error line levels. The levels are the number of images out of the 500 images of the hand shape sequences where the tracking fails

that will be attained with most real images. According to figures 6.16 and 6.17 a suitable decimation ratio of the hand shape tracking would be roughly 0.3. For the watch tracking it would be 0.1. This suggests that the ratio of references is not a measure that could be directly linked with the level of robustness of the tracking. Consider the ratio of 1% for the hand stencils, three of these stencils are shown in the last row of figure 6.23. One of the stencils has only one element, that correspond to roughly 90 other stencils. At that level of decimation, the hand tracking has become meaningless as confirmed by the last row of the histograms of figure 6.25 that shows the number of votes that stencils get for three different images of the sequence. In contrast, at the same level of decimation for the watch, tracking is still feasible as shown by the last row of figures 6.34 and 6.36. This is again simply due to the fact that the watch shape has many more features than the hand shape.

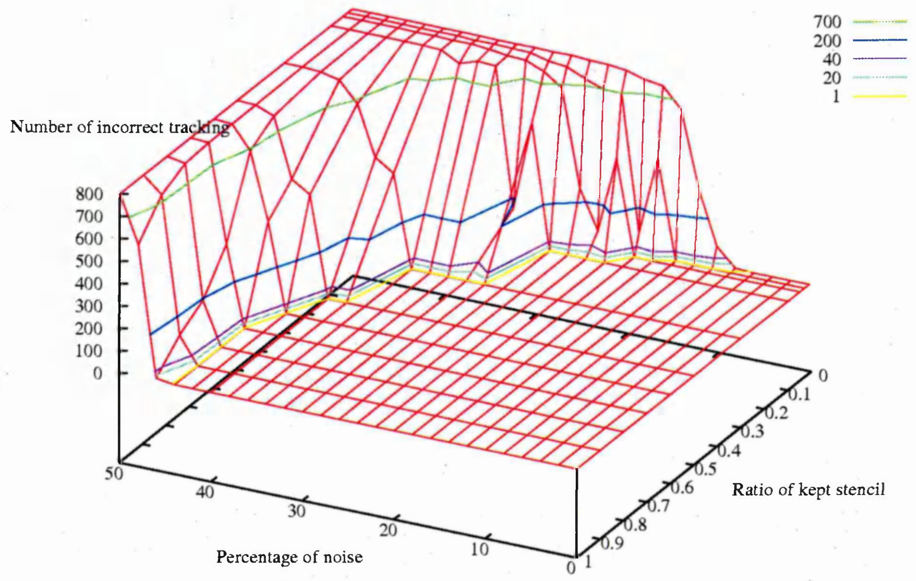


Figure 6.18: The z-axis corresponds to the number of images where the tracking result differs from the ground truth. The ratio of kept elements of the stencil is actually the ratio of the number of references listed in the 2-D array corresponding to the stencil elements kept. The colour lines outline error levels. This graph corresponds to the watch shape sequences

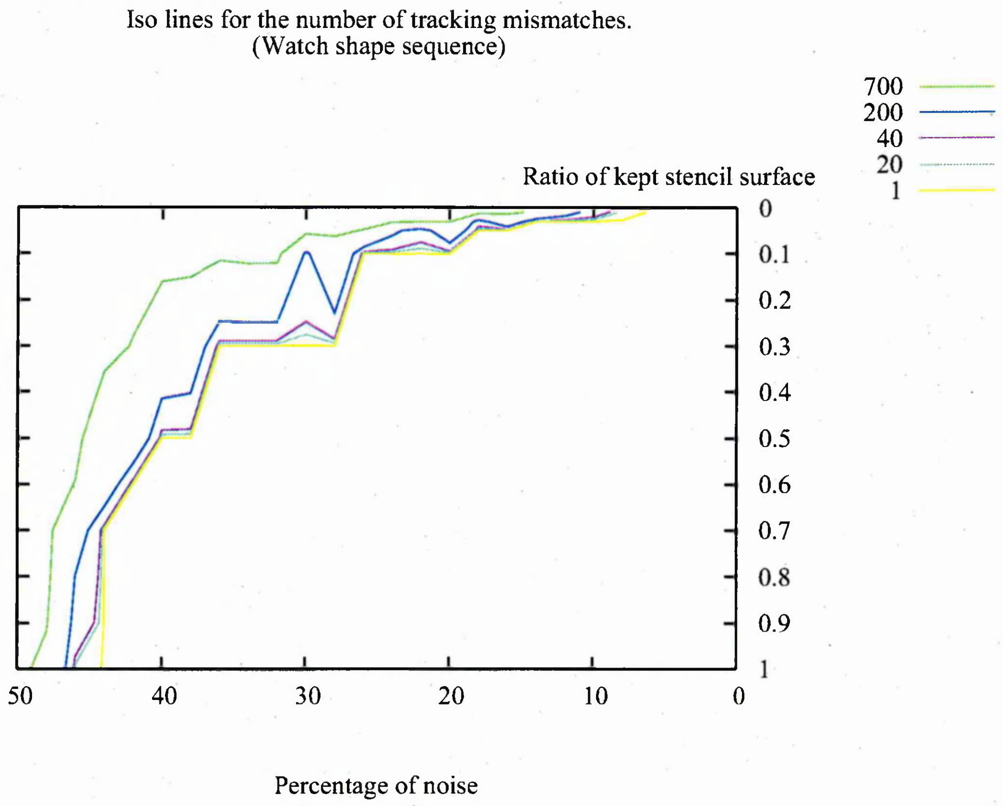


Figure 6.19: The same graph as in figure 6.18 viewed from the top and with just the error line levels. The levels are the number of images out of the 800 images of the watch shape sequences where the tracking fails

It was first believed that a good criteria to stop the decimation of the stencil would be a threshold on the number of points of a stencil. However, when determining a suitable level of decimation relative to a given noise ratio it was realised that it did not. Let us consider the noise level at 20%. As mentioned above, the desirable corresponding decimation ratio for the hand shape and watch shape were respectively 0.1 and 0.3. By analysing a few stencils on figures 6.22, 6.23, 6.33 and 6.34 with the corresponding level of decimation it can be observed that the number of stencil elements does not fully explain the tracking performances relative to salt and pepper noise. The number of stencil elements per surface area might be a better indicator for characterising a certain level of robustness to a given level of salt and pepper noise. If this is the case, it may allow the automatic selection of a threshold level to stop the stencil decimation, given a level of desired robustness, defined using the salt and pepper noise level.

Looking at the first column and first row of Figures 6.20 it can be seen that for the hand shape the maximum number of references, $|\text{array}(p)|$, is around 180. Half of the elements of about 4000 elements each refer to less than 40 compatible transformations. The second column shows the histogram of the number of elements given the size of the reference list. The distribution may be better understood this way. It can be seen that the shape of the histogram changes significantly when stencils are reduced. The largest number of references an element of the array can have reduces quite steadily and significantly with the reduction of the stencil size. Tracking wise, the number of operations are reduced since for a given feature point, fewer references receive votes. The histograms of the second column also show that the wide majority of the elements of the array does not refer to any transformation. The first column is the cumulative histogram of the histogram of the second column, starting from elements having 1 reference. It therefore shows the maximum number of votes that could be obtained if all points having less than n references were selected. When tracking, not all of these points are selected. However it outlines the impact of points having a large number of references. Looking at the first row of figure 6.20, we mentioned that half of the elements, *i.e.* about 2000 for the hand shape, each contained less than 40 elements. The histogram on the first row of the third column shows that they are roughly responsible for a sixth of the maximum number of votes that could be generated. In other words a minority of the elements of the look up table array are responsible for most of the time consumed during the tracking phase. The stencil reduction allows this to change: looking at the fourth row of figure 6.20 it appears that the maximum number of elements is about 35 and that the maximum level of votes that could be generated is approximately 50000. When compared with the first row, it roughly corresponds to twice as many as the maximum number of votes that could be generated by half of the elements that were referencing about 40 stencils and less.

Figure 6.26 shows the average of the number of elements over all stencils for a stencil that is not overlapped by its most overlapping stencil. When a stencil is intersecting the shape, because they share a large number of points, its most overlapping stencil is likely to be one of the other stencils that obtains the highest number of votes. Therefore, this value can be considered as an indicator of the robustness of the stencil estimator. Figure 6.26 shows the total robustness indicator *i.e.* the minimum over all stencils of the number of elements of a stencil that are not overlapped by its most overlapping stencil. Both curves do not differ significantly, indicating a certain homogeneity from one stencil to another. Since only translations of the shape were considered and the sub-transformation space size was such that the stencil shape and the template shape had the same number of elements this homogeneity can be interpreted as follows: the most overlapping stencil for a given stencil was one of the stencil slightly translated from the given stencil. This pattern was observed many times, except for some of the stencils corresponding to the boundary translation values of the transformation space. For the boundary values of the transformation space, the most overlapping stencil, when it did not correspond to the same translation pattern, intersected about the same number of elements of the stencil.

These observations are similar for the watch stencil estimator as shown by figures 6.37 and 6.38. However, the robustness indicator value is much higher. Nevertheless, the data is insufficient to be able to correlate robustness with this proposed indicator.

Figures 6.28, 6.29,6.30, 6.39, 6.40, 6.41 sum up time measurements for the different tracking sequences. Each measure corresponds to the average time over, respectively, the 500 images of the hand shape sequences and the 800 images of the watch shape sequences. The average time appears to vary almost linearly versus the rate of noise and the ratio of kept references.

Number of elements of the reference array having less than n position references	Number of reference of the reference table having n elements in their list	Maximum number of votes that can be produced by the elements having less than n position references

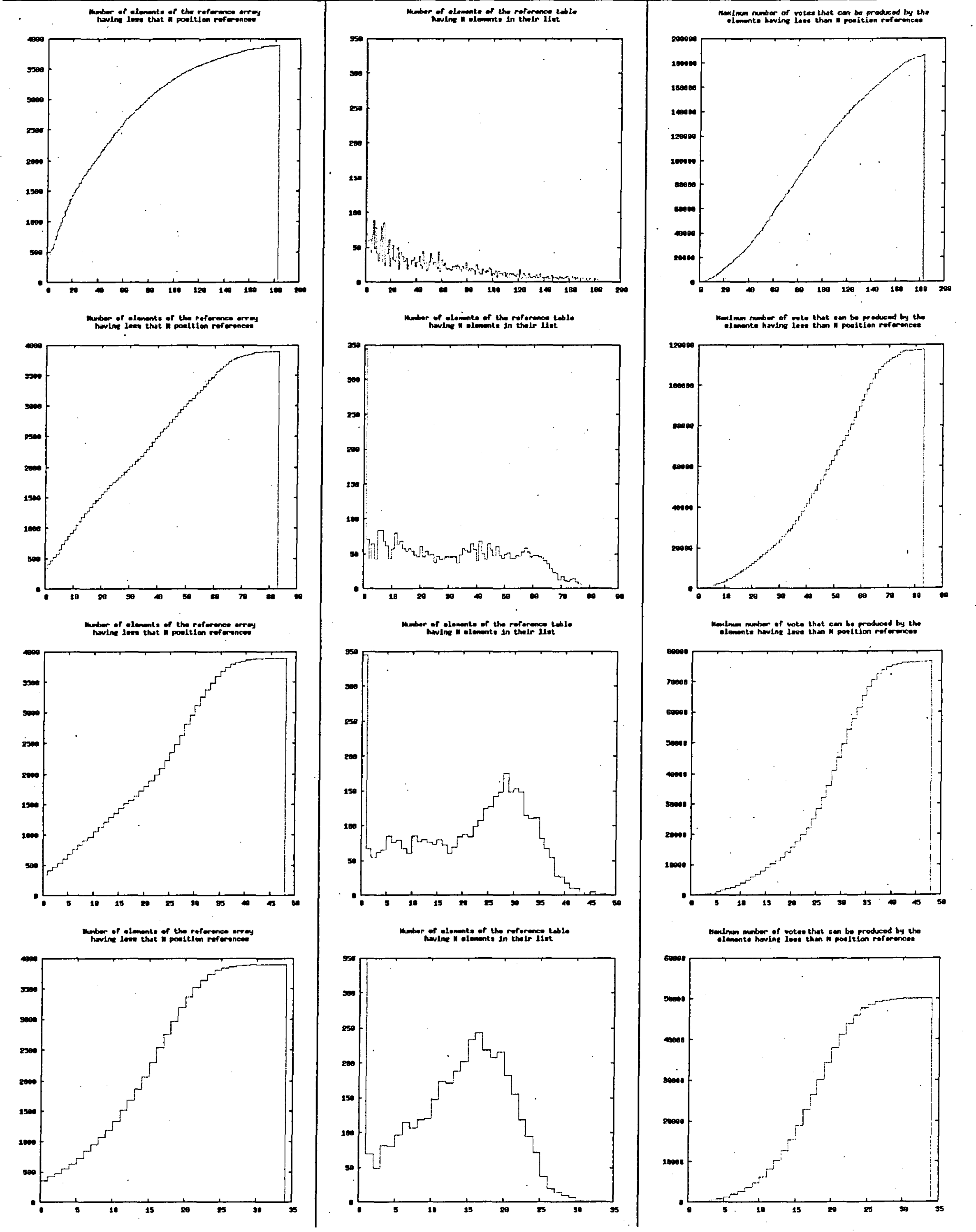


Figure 6.20: Histograms characterising the 2-D array containing stencils generated off-line for the hand shape. Each row corresponds to a different tuning of the area size of the stencils. These are 100 %, 56 %, 32 % and 18% of the references respectively

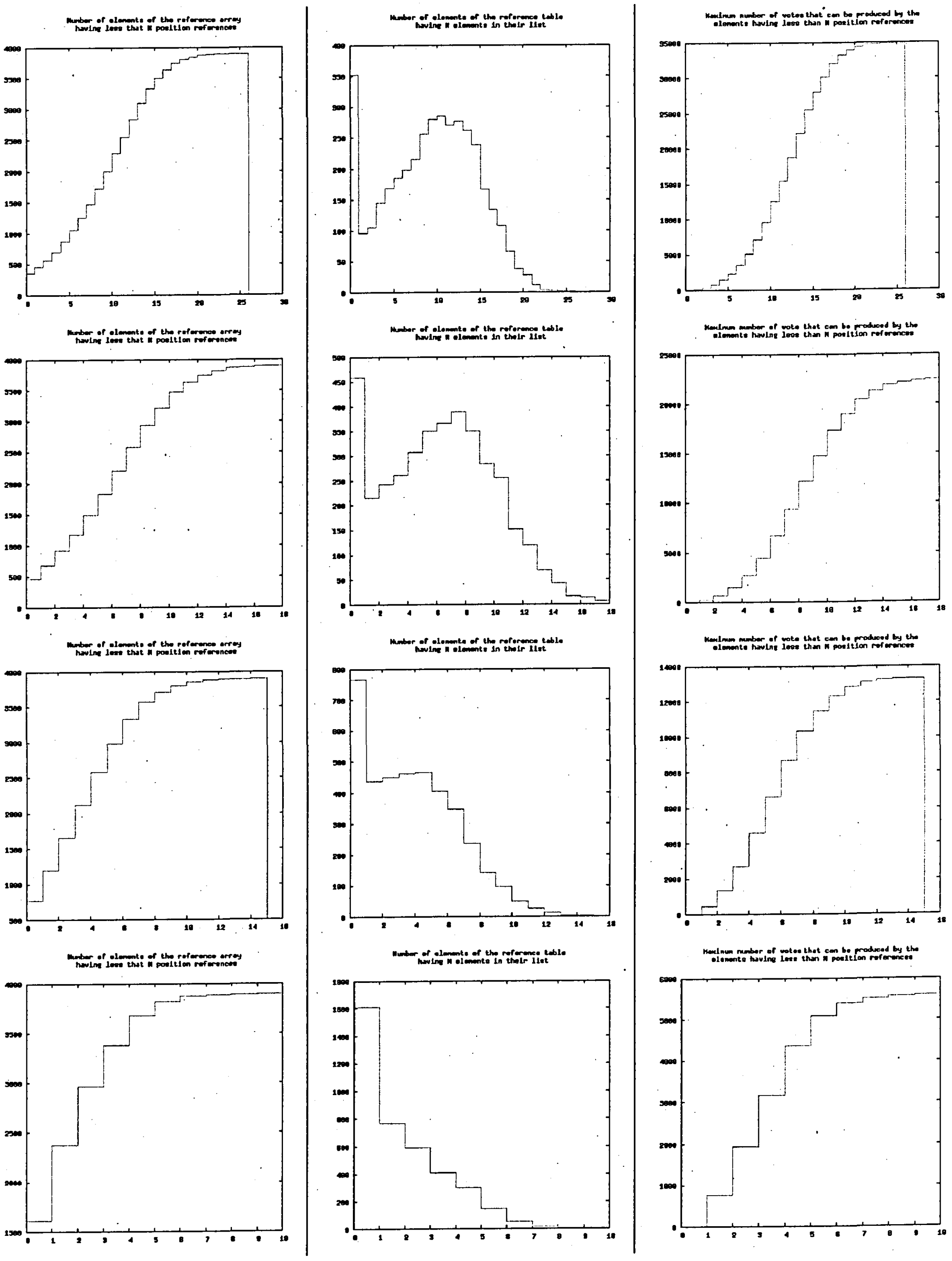


Figure 6.21: Histograms characterising the 2-D array containing stencils generated off-line for the hand shape. Each row corresponds to a different tuning of the area size of the stencils. 11 %, 6 %, 3% and 1% of the references respectively

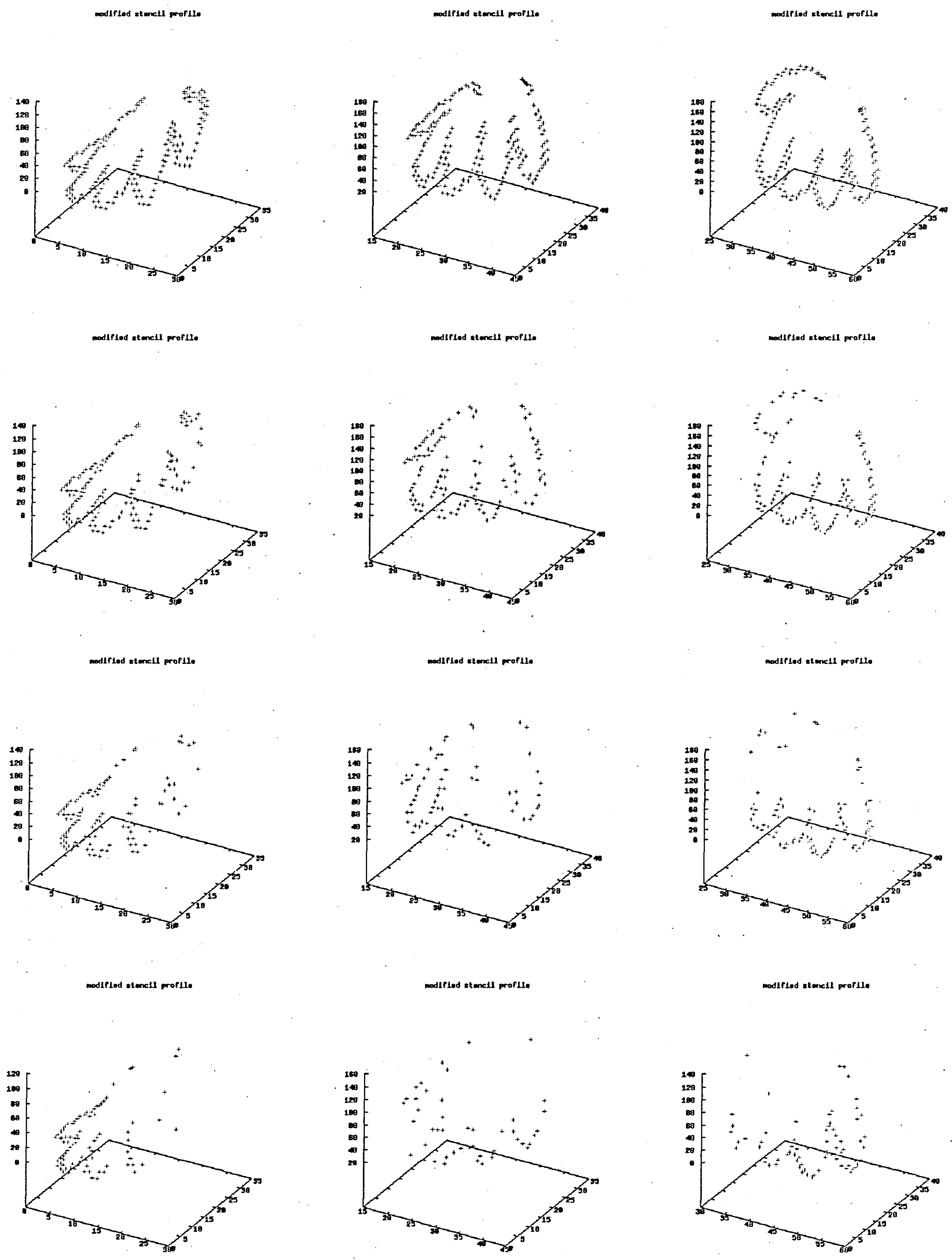


Figure 6.22: Stencils for the hand shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to those of figure 6.20: 100%, 56%, 32%, 18%

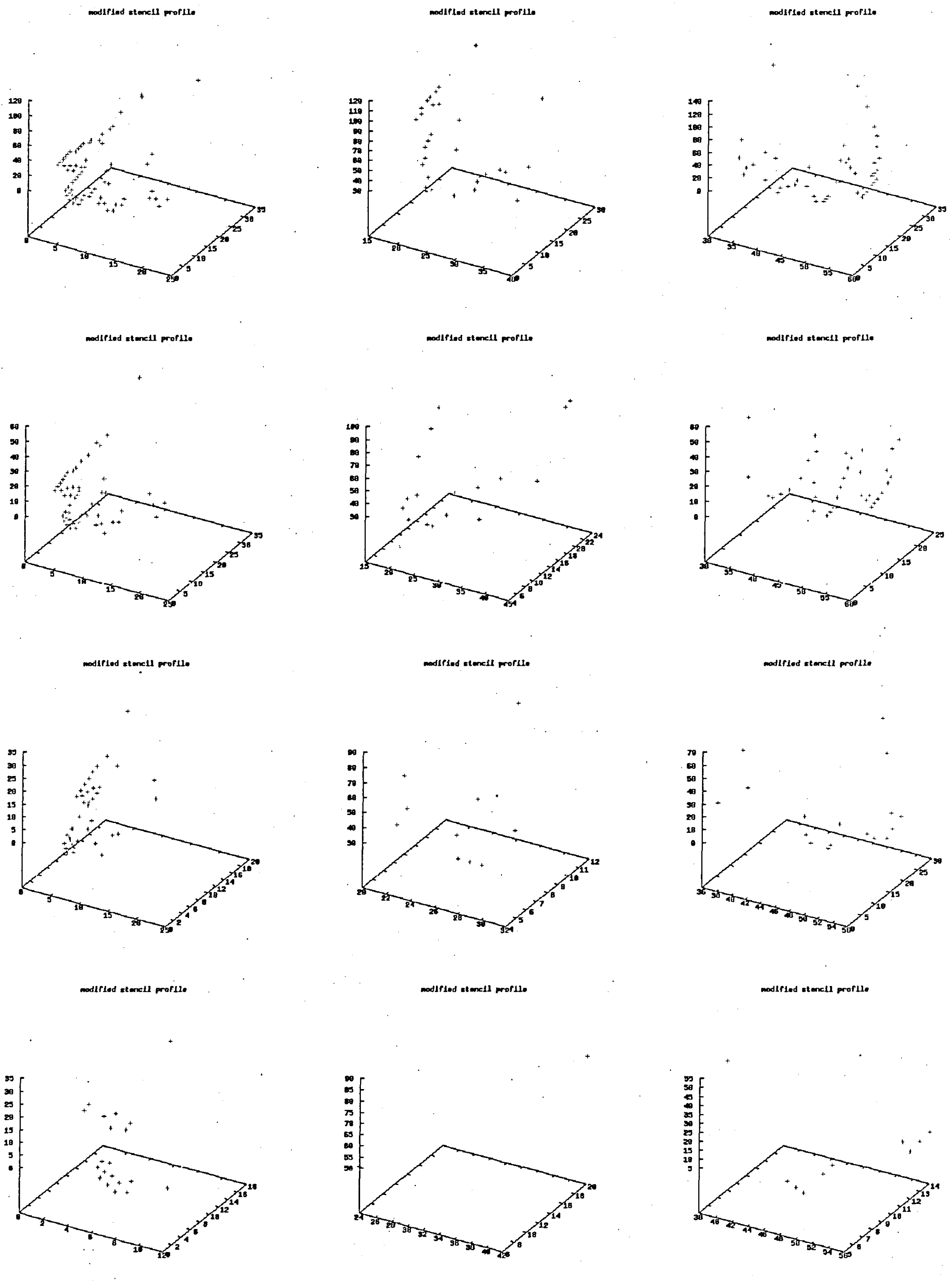


Figure 6.23: Stencils for the hand shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to those of figure 6.21 (11%, 6%, 3%, 1%) and each column a different stencil

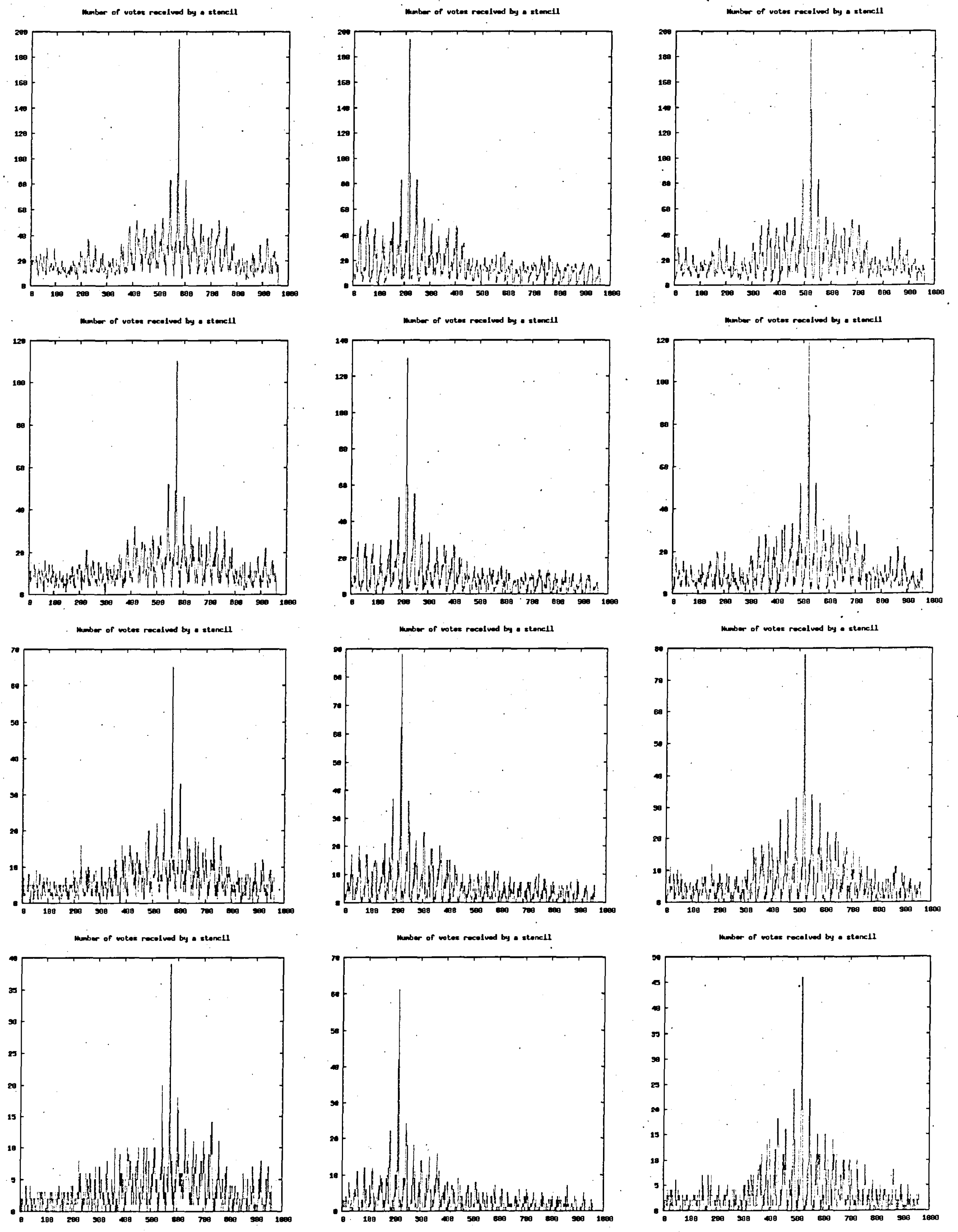


Figure 6.24: Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to those of figure 6.20. Each column represents a different image from the hand image sequence

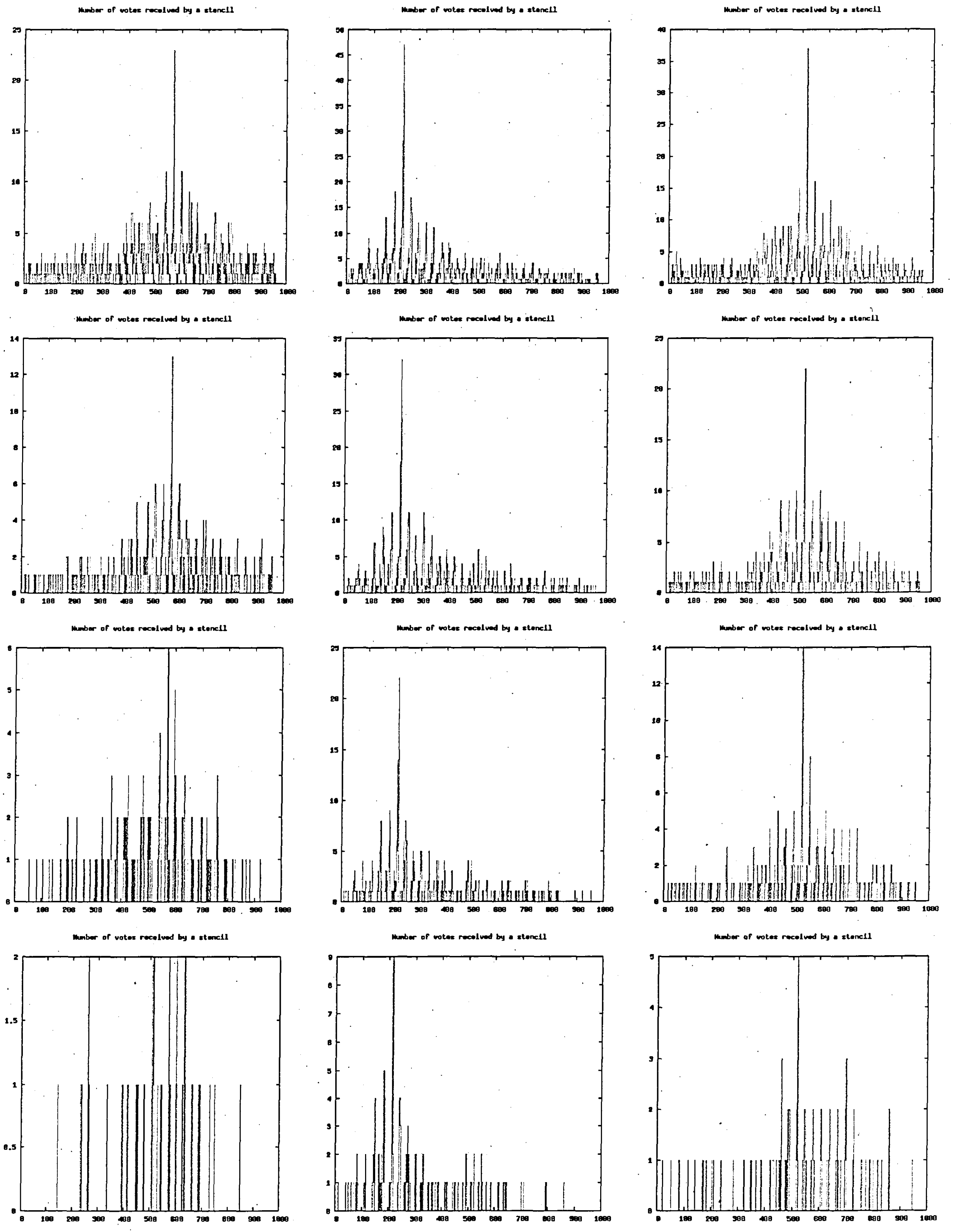


Figure 6.25: Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.21. Each column represents a different image from the hand image sequence

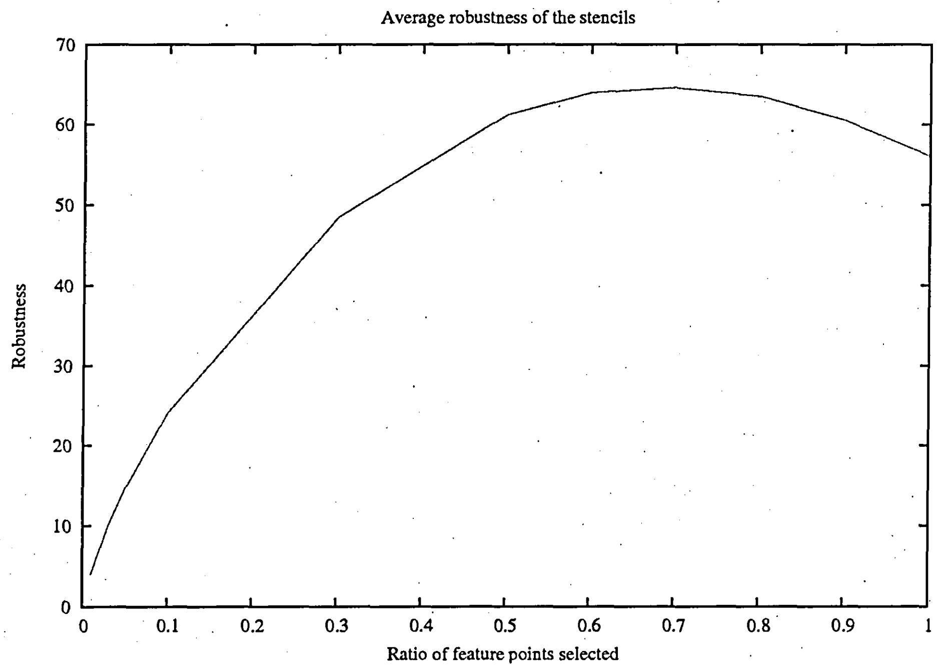


Figure 6.26: What has been termed average robustness is, in fact, the average difference of, the number of elements of a stencil, and, the number of overlapping elements of its most overlapping stencil

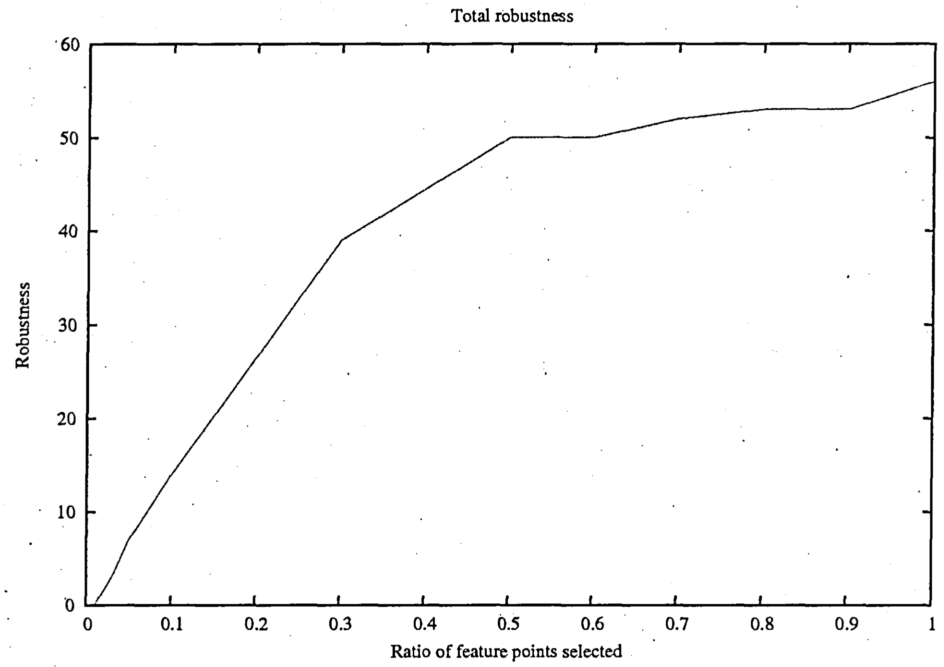


Figure 6.27: By total robustness we refer to the minimum, for all stencils, of the robustness such it is explained in figure 6.26

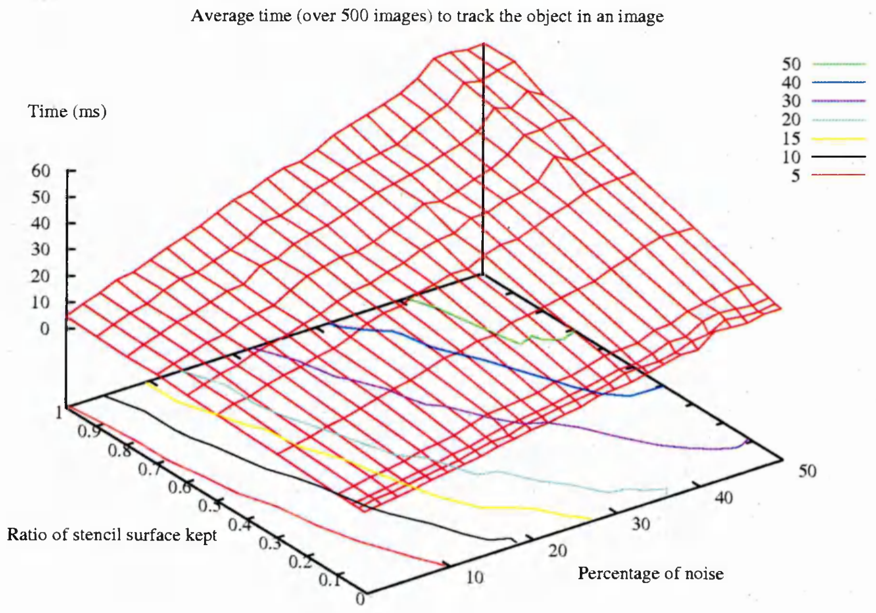


Figure 6.28: Time taken to track a shape versus the level of noise and the decimation ratio of the stencils for the hand image sequences

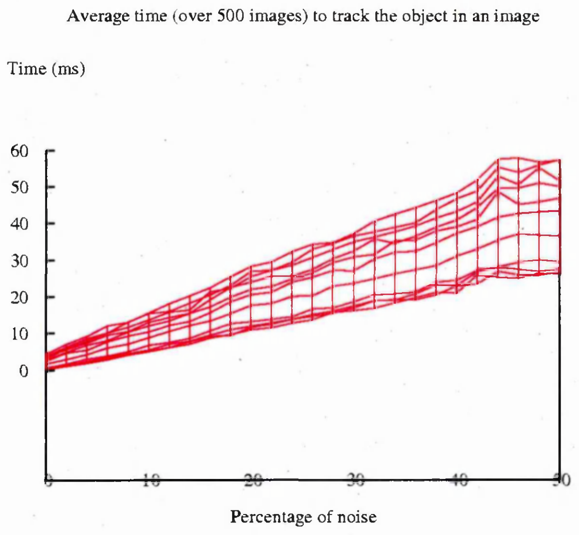


Figure 6.29: The graph of figure 6.28 from a different viewpoint. This shows how the level of noise affects the tracking time

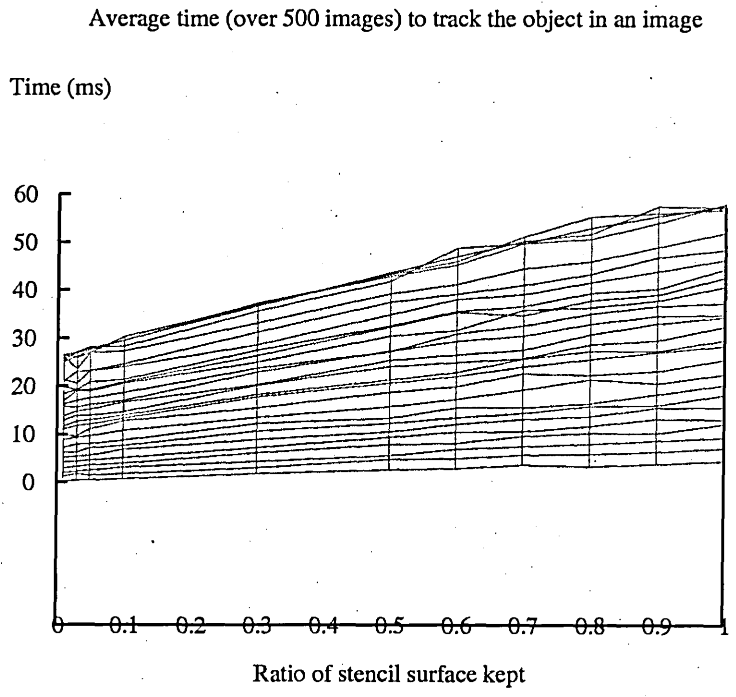


Figure 6.30: The graph of figure 6.28 from a different viewpoint. This shows how the stencil decimation ratio affects the speed performance of the tracking

6.2.2 Comparison with different similarity methods using an exhaustive search

For comparison purposes, we implemented the stencil estimator using algorithm 3.

Algorithm 3: Tracking using the stencil estimator by moving the shape across the region of interest.

```
foreach position near to the previous object location do
    foreach element of the shape of the object do
        if the shape element corresponds to an image feature then
            increase the number of votes for the current position by one
return the position that has the maximum number of votes
```

A correlation algorithm would iterate in the same way for a small region of interest[1], however this algorithm differs in two ways: firstly, instead of using a rectangular shape surrounding the image for computing the correlation the stencil estimator can use the features of the shape only, thus reducing the number of operations needed. Secondly, the measure used is not the correlation measure but the number of matching features to the shape. Since it is faster to evaluate the number of matching features than to correlate the points (which involves various multiplications), and less points are considered, only the shape points and not all those in the rectangular area surrounding the shape, this measure is faster (see table 6.1).

Tests for the watch sequence and the hand sequence were realised on two different computers and therefore cannot be directly compared in terms of speed (the computer used for the watch sequence was between 2 to 4 times slower). While no stencil reduction is needed to track the hand shape in real-time, it offers a marked improvement for the watch shape that contains many more points than the hand shape.

Table 6.1 also shows the benefits of using the bounded Hough transform algorithm. This difference in speed can be explained by two factors: the two embedded for loop calculations are factorised into the pre-processing stage. Also, when the tracking is performed with the bounded Hough transform, votes are increased only for image feature points. It is therefore equivalent to omitting the check as to whether a shape point corresponds to an image feature when no feature is present

[1]If the region of interest becomes large it is more efficient to perform the convolution of the image and the template image in the frequency domain (using the fast Fourier transform).

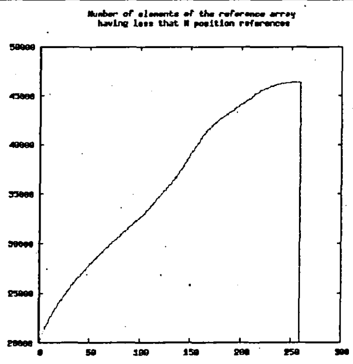
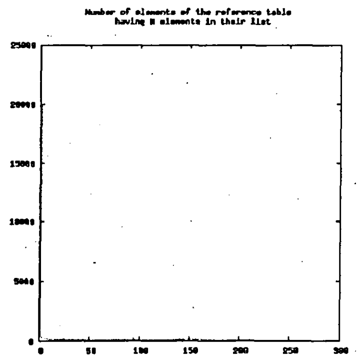
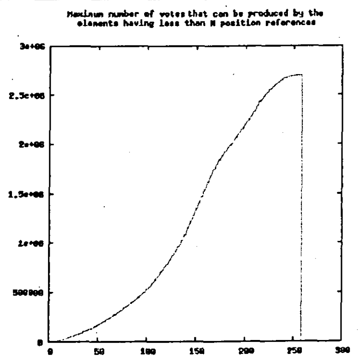
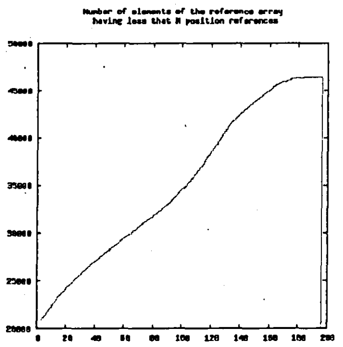
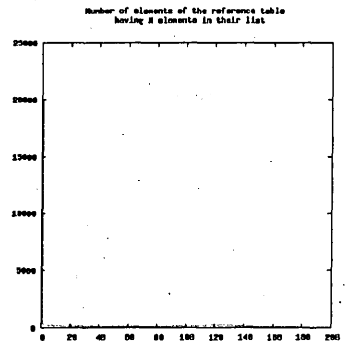
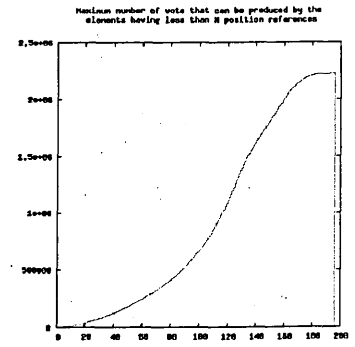
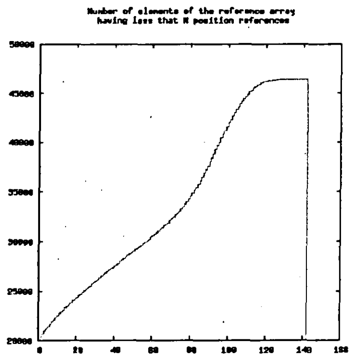
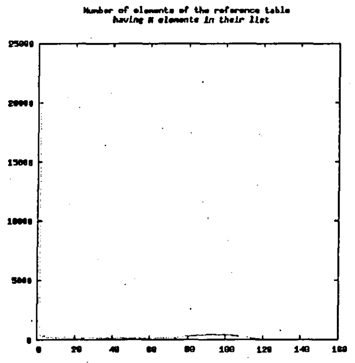
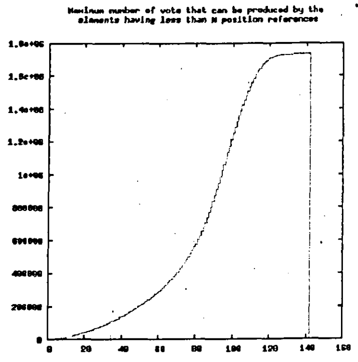
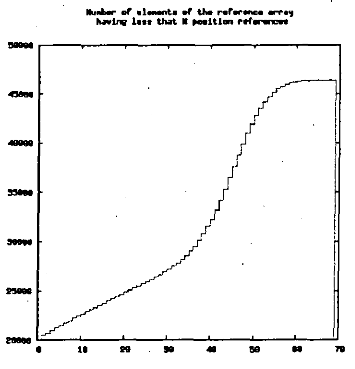
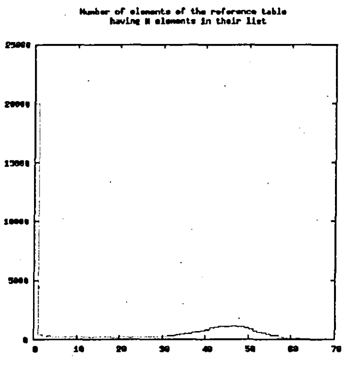
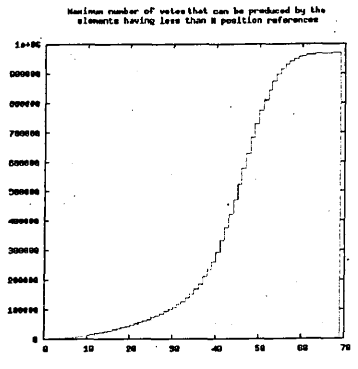
Number of elements of the reference array having less than n position references	Number of reference of the reference table having n elements in their list	Maximum number of votes that can be produced by the elements having less than n position references
		
		
		
		

Figure 6.31: Histograms characterising the 2-D array containing stencils generated off-line for the watch shape. Each row corresponds to a different tuning of the area size of the stencils. These are 100 %, 80 %, 60 % and 30% of the references respectively

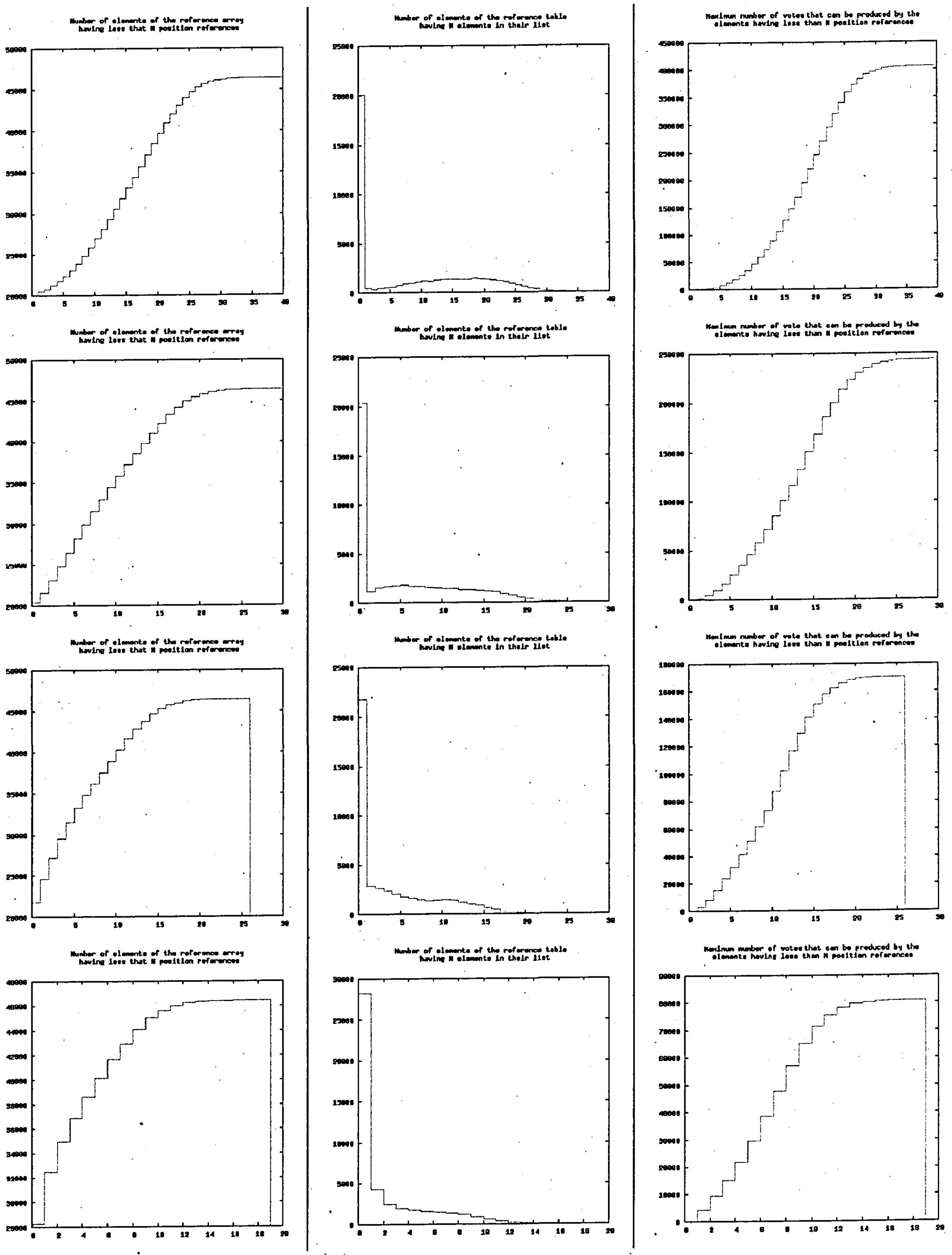


Figure 6.32: Histograms characterising the 2-D array containing stencils generated off-line for the watch shape. Each row corresponds to a different tuning of the area size of the stencils. These are 10 %, 5%, 3% and 1% of the references respectively

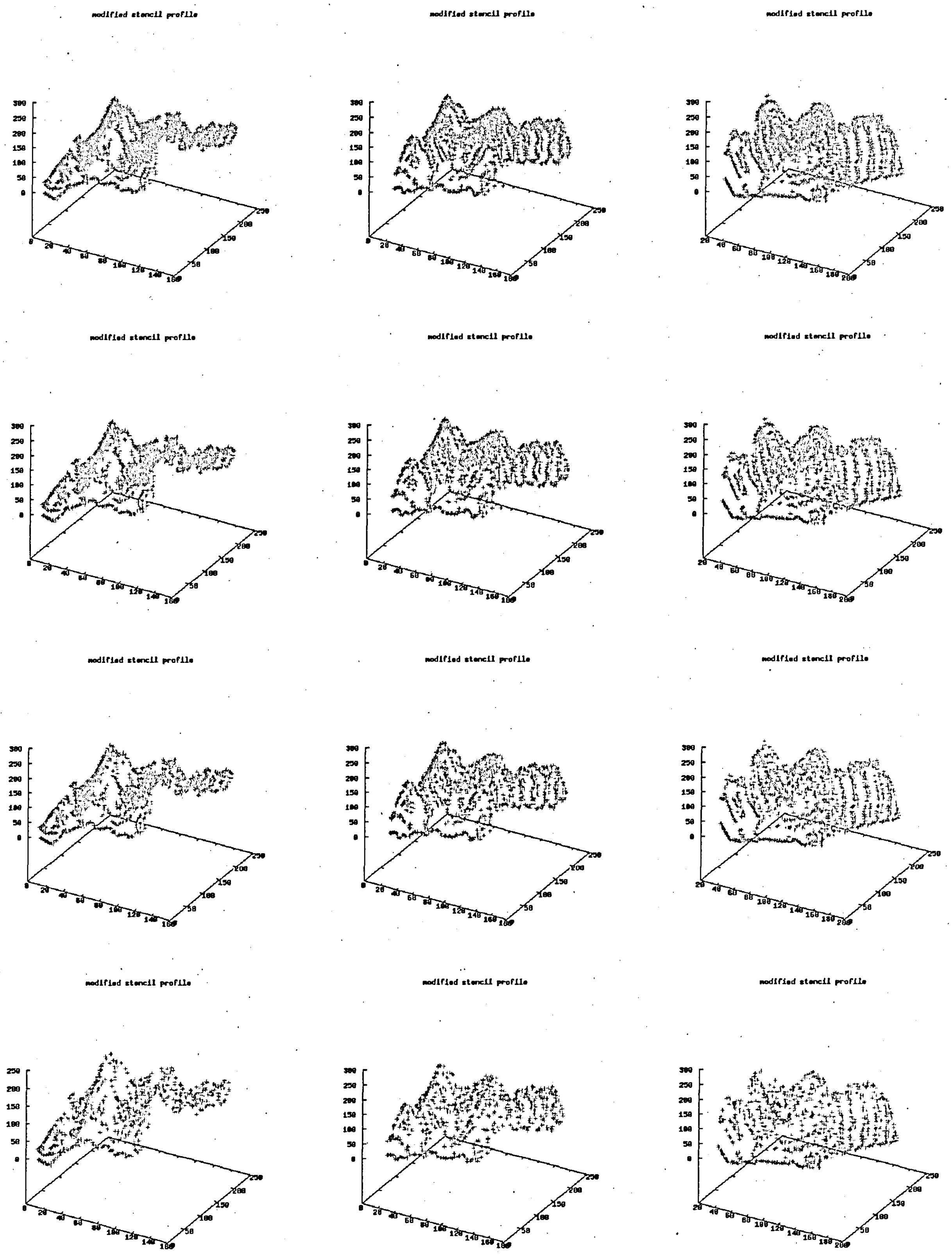


Figure 6.33: Stencils for the watch shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to figure 6.31

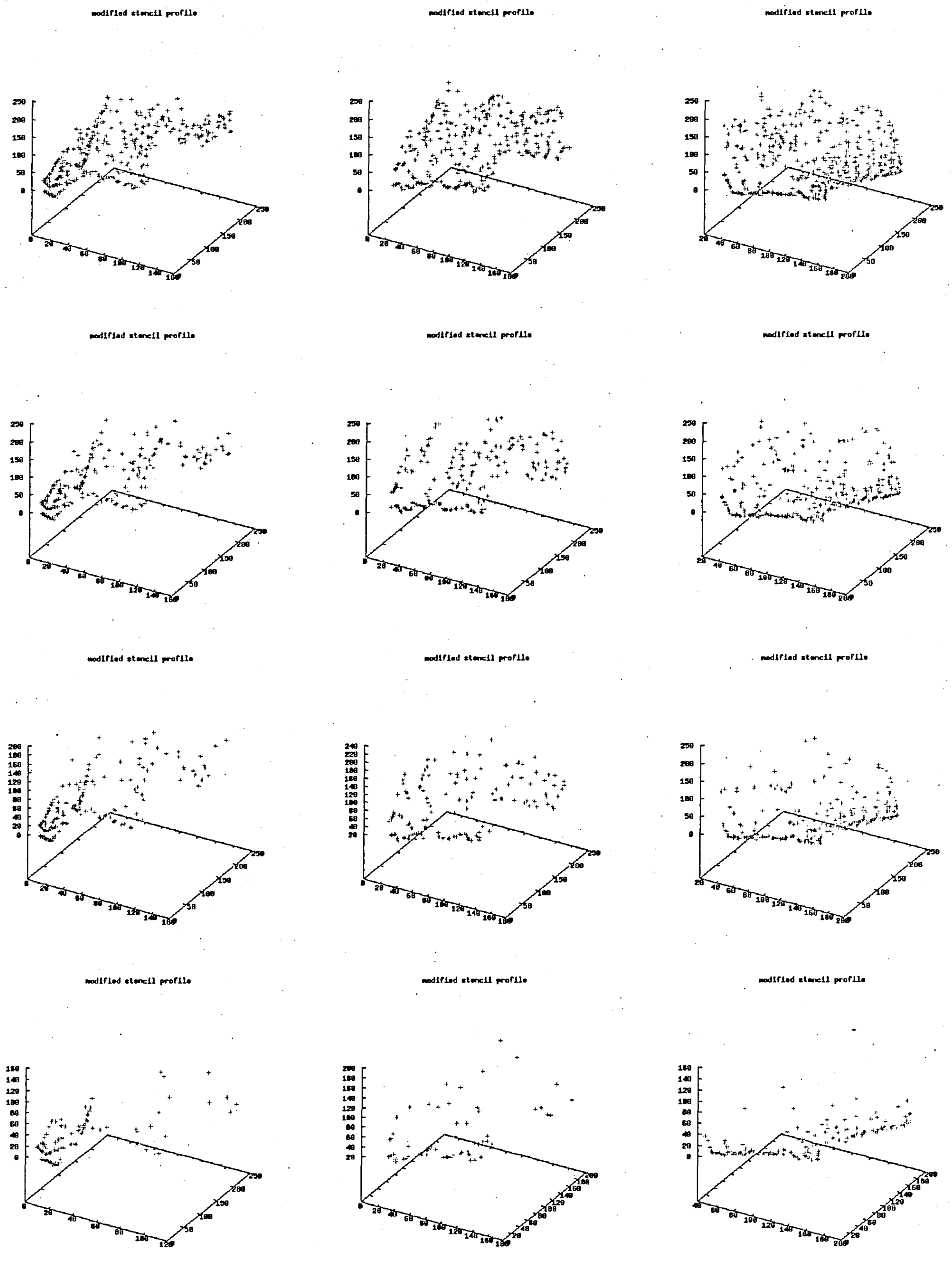


Figure 6.34: Stencils for the watch shape. The z-axis represents the number of overlapping stencils. Each row represents a different stencil decimation level corresponding to figure 6.32

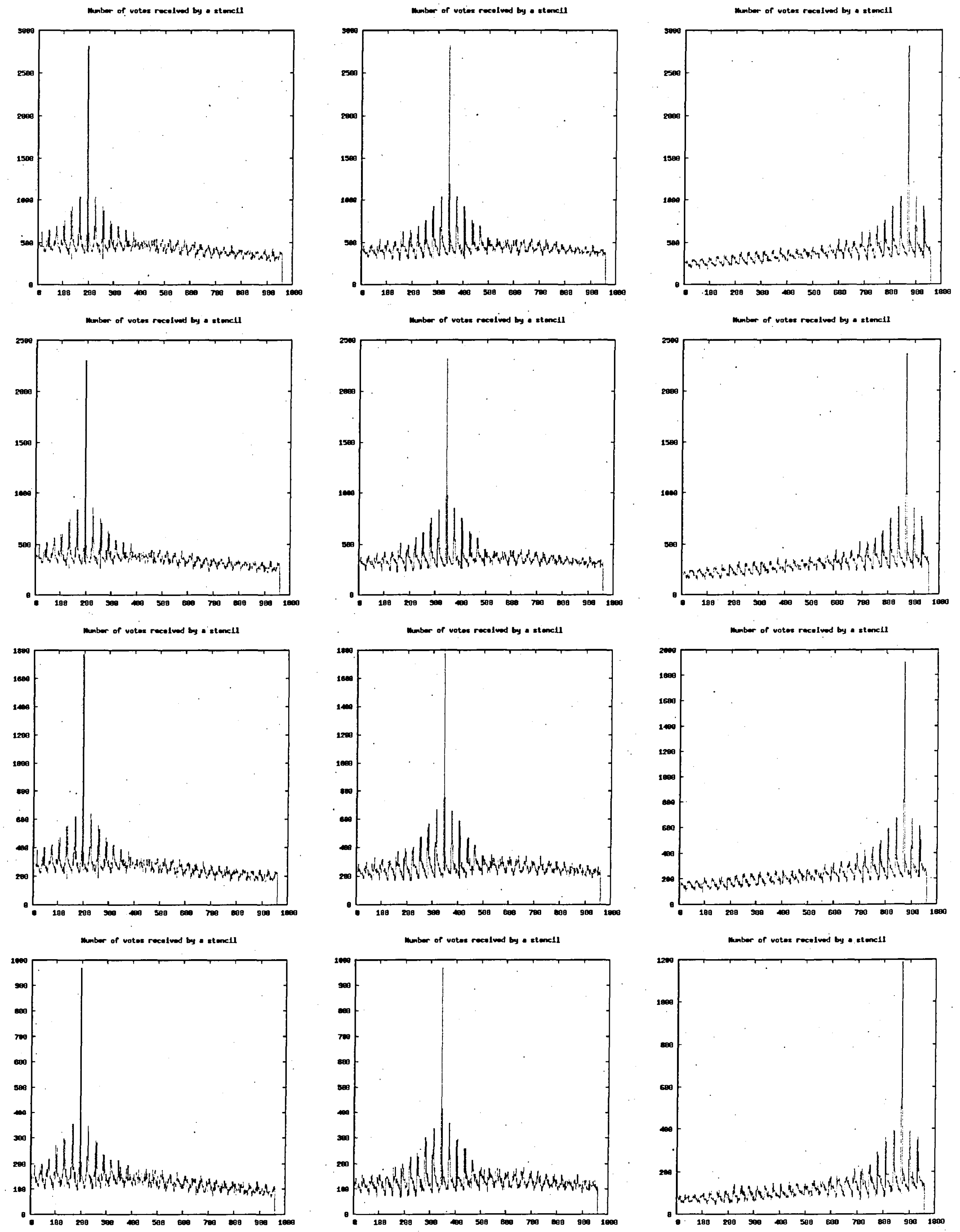


Figure 6.35: Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.31 . Each column represents a different image from the watch image sequence

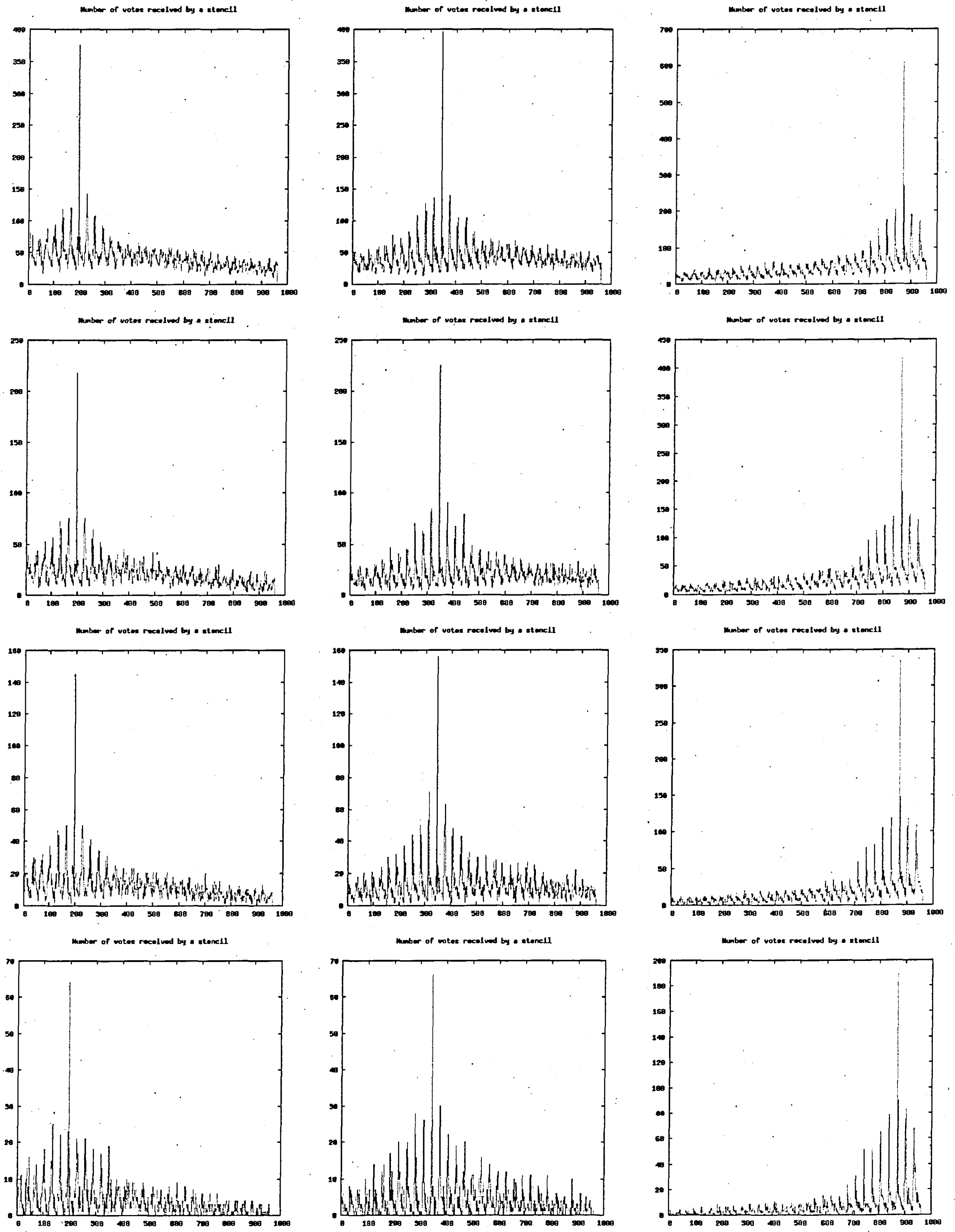


Figure 6.36: Number of votes for each stencil. Each row represents a different stencil decimation level corresponding to figure 6.32 . Each column represents a different image from the watch image sequence

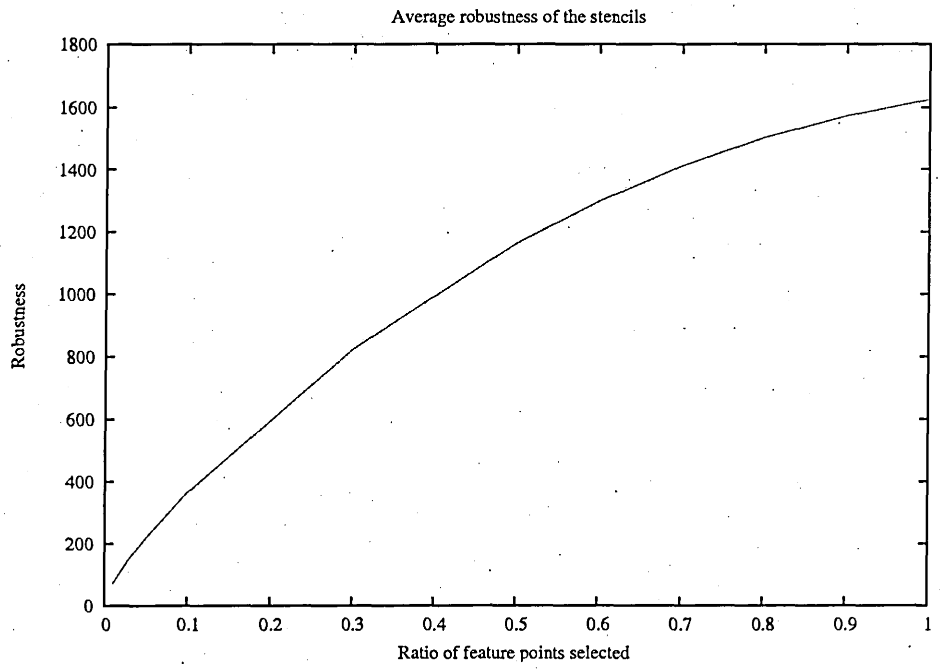


Figure 6.37: What has been termed average robustness is, in fact, the average of the difference of, the number of elements of a stencil, and, the number of overlapping elements of its most overlapping stencil

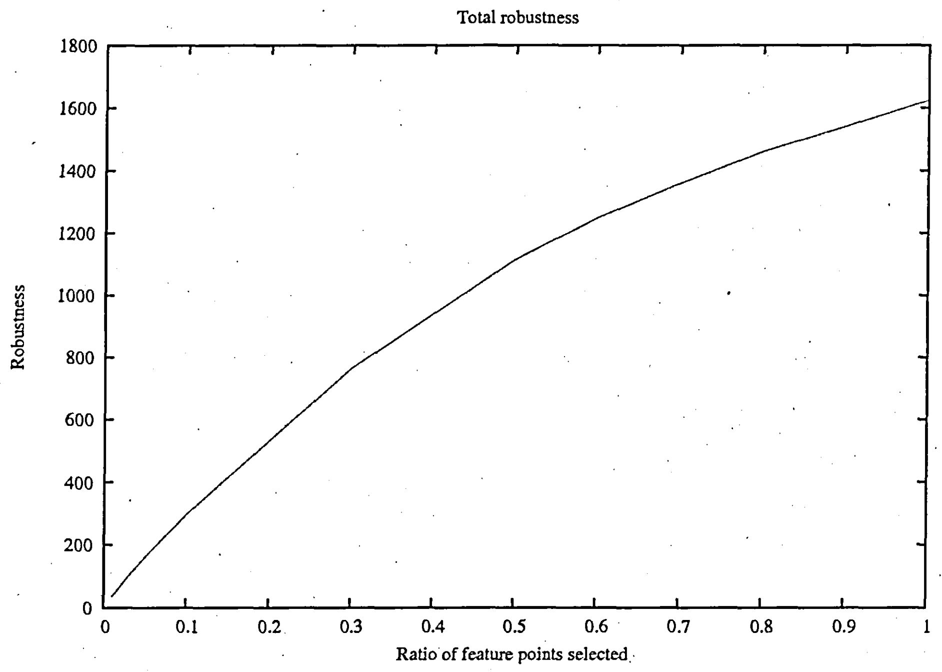


Figure 6.38: By total robustness we refer to the minimum, for all stencils, of the robustness such it is explained in figure 6.37

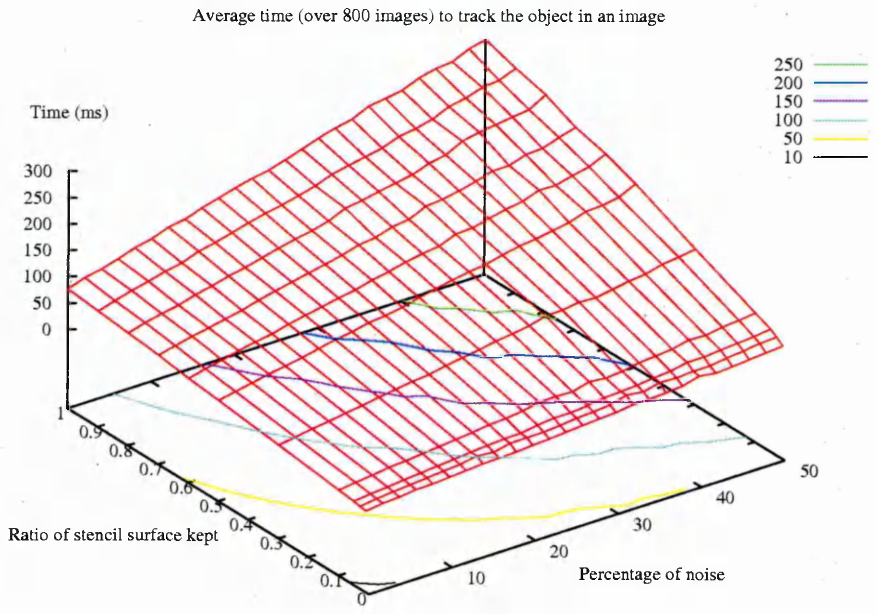


Figure 6.39: Time taken to track a shape versus the level of noise and the decimation ratio of the stencils for the watch image sequences

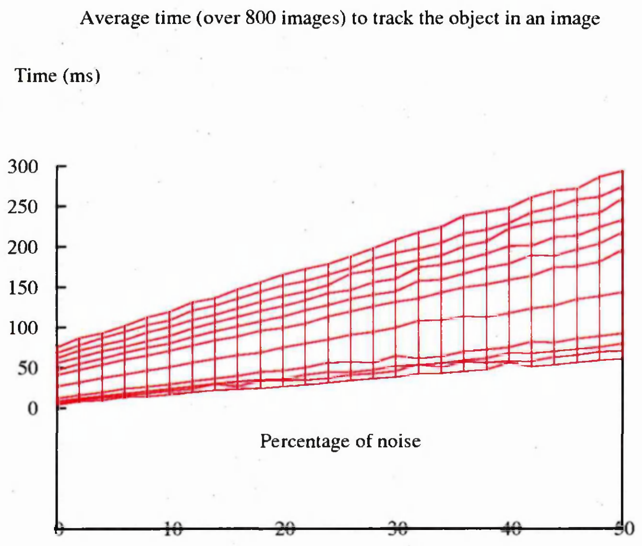


Figure 6.40: The same graph as in figure 6.39 but with a different viewpoint. This shows how the level of noise affects the tracking time

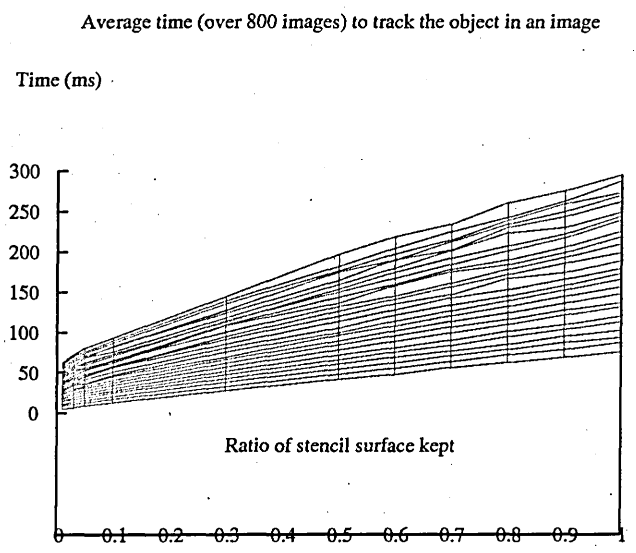


Figure 6.41: The same graph as in figure 6.39 but with a different viewpoint. This shows how the stencil decimation ratio affects the speed performance of the tracking

Image sequence description	**Hand shape sequence** (194 feature points)	**Watch sequence** (2812 feature points)
Best time using cross correlation with a rectangular template	1967 ms (1x) (windows size 50x50)	27931ms (1x) (windows size 161x213)
Best time using cross correlation. The template have the shape of the object	158ms (12.4x)	2050ms (13.6x)
Best time using using the stencil estimator	142 ms (13.8x)	1705ms (16.4x)
Average time using the stencil estimator with the bounded Hough transform algorithm (full stencil)	5.5ms (358x) average over 500 images	75.4ms (370x) average over 800 images
Average time using partial stencils 50%	3.5ms (562x)	41ms (681x)
Average time using partial stencils 10%	2.3 ms (855x)	12.4ms (2252.5x)
Average time using partial stencils 1%	Not relevant: some stencils don't even have a point	4.6ms (6072x)

Table 6.1: Comparison of the speed of different algorithms to track an object translated in 2-D in its surrounding.

in the corresponding image position.

If the reader wishes to carry more tests, a test bed, developed in C++, has been provided on-line as an example application within the Mimas library (version 2.1). Facilities are also provided to generate image sequences as well as to display the results directly into diagrams.

6.2.3 Summary Comparisons with the cross-correlation similarity measure show the superiority of the stencil estimator in term of speed performances, especially when the stencils are used in conjunction with the bounded Hough transform: a speed-up of a factor of 358 and 370 respectively for the hand and the watch shapes. These performances can be further improved by decimating the stencils by a factor of 2 to 10 depending of the shape of the object and the robustness requirement of the application.

Interestingly the stencil estimator seem to match or outperform the human vision system. Psycho-physicists might be interested to further investigate this maybe illusory but striking correlation, especially when considering that the parallel gathering of information is not structurally incompatible with neural networks. However the extent of this study is lacking evidence to determine whether this is a coincidence or not.

Future research

Having a criteria to decide automatically when the stencil decimation has to stop would remove the need for manual tuning. The density of retained points of the stencil might provide a criteria for a given level of salt and pepper noise resistance. Histograms presented in figures 6.20, 6.21, 6.31 and 6.32 could also be useful to determine a criteria to stop the stencil reduction.

The speed of the tracking is enhanced by the reduction of the stencils' area while remaining robust. However, more research is needed to determine an optimal way to reduce the stencils' area. For instance the weight of the stencils elements could be determined in a different way.

For detection of the shape in a whole image we propose a hierarchical approach. Large stencils could be used to coarsely determine the position of a shape followed by successive refinements. Determining the number of stages that optimally enhance the tracking performance is one of the issues that needs to be tackled. At this point it also appears that the density of features present in an image can be a useful cue to locate an object, the bounded Hough transform can be adapted to practically measure this density.

Similarly it is possible to separate the translation component of the state space from other transformation such as scale, rotation and shear. One possible way to implement this would be to first make a stencil able to encompass the maximum size of the object and then filter the image to establish the regions of interest where the shape could be located; this would be followed by a second stage where the stencilled Hough transform could be operated on these pre-determined region of interest. However, in images where features are dense this may not work since any part of the image would potentially be a region of interest. Designing a stencil that better characterises the presence of a shape in a region of interest might solve this issue.

6.3 Testing the particle filter on the micro-pipette tracking sequence

Figures 6.42 to 6.49 illustrate scene images and their corresponding *pdf* based on the edge template correlation. Figure 6.42 and 6.43 illustrate the optimal case where particles are clustered around the micro-pipette tip, the resulting *pdf* is almost uni-modal. Figure 6.44 illustrates the case where particles are spread out, the resulting *pdf* (figure 6.45) is multi-modal because of the influence of background clutter. The multiple modes are subsequently resolved and the pipette tip is correctly located by using further measurements to evaluate modes. Figure 6.46 illustrates the case where the correlation of the edge template results in most of the peak points around the tip of the pipette. But the particles are spread out and subsequent measurements incorrectly select the mode that is far away from the actual location of the tip. Improving the multivariate measure might correct this problem. Figure 6.48 and 6.49 illustrate yet another scene where the background clutter and change in scene lighting give rise to multiple modes. However, in this case the multiple modes are correctly resolved by additional measurements.

The graph shown in figure 6.51 shows the plot of the actual location of the micro-pipette (manually traced) and the corresponding tracked location of the micro-pipette tip. The histogram of figure 6.52 shows the repartition of all frames classified according the accuracy of the software tracked location. To give an idea of the accuracy, the actual location has been manually compared with an average error of less than 3 pixels despite all the constraints. The micro-pipette width is 48 pixels. The graph shows that in 9 frames out of 329 frames the actual location and the probable tracked position differs by more than 21 pixels.

The graph shown in figure 6.53 illustrates the same tracking scenario but with the tracked locations plotted only when the measure is above 0.8 (values ranging from 0 to 1). This graph, by showing that all the tracked location having a high value measure associated are the expected ones, illustrates that a certain threshold level allows to be confident that the object is accurately located. It can also be seen that for dozens of frames, no tracked location information is available. Trusted tracked locations can typically be used to partially re-initialise the particle filter.

In a scenario, where clutter comes from the presence of multiple instance of an object in the image, partial reinitialisation cannot be applied since measures of other object instances are almost identical to the tracked object. However, previous dynamic of the object can be incorporated to the particle filter with the above mentioned method.

Further tests of the implementation of the particle filter have been carried out, appendix C presents and discusses the adaptations needed for these tests and their results.

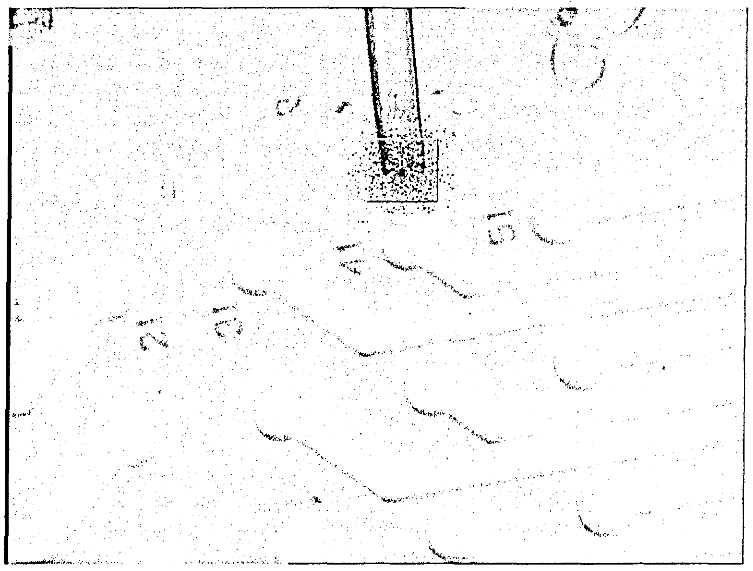


Figure 6.42: The template is shown on the top left corner. The small black dots are the particles. The centre of the square is the tracked location.

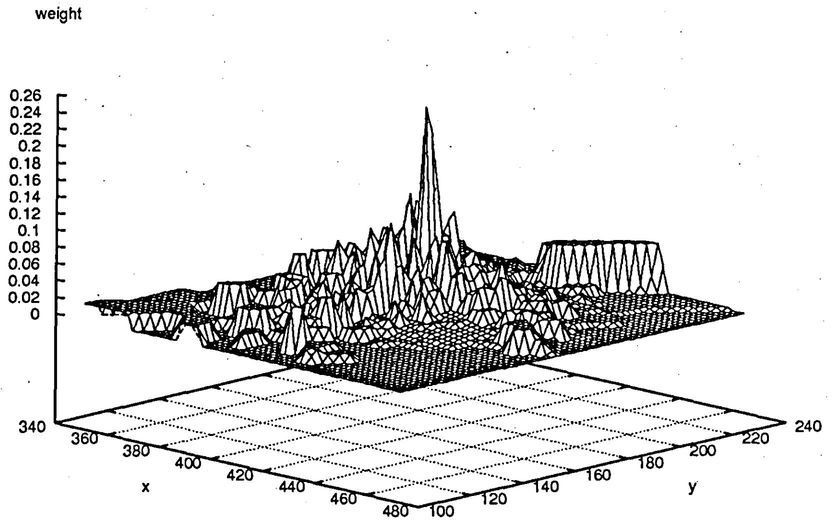


Figure 6.43: The weight measures for this graph and the subsequent graphs represents edge correlation measure. This is an ideal case where the measure gives a unimodal *pdf.* x and y axis are the pixel coordinates of the image point measured. It can be observed that correlation measure is well localised which is a disadvantage for particle filters that are likely to sample the tracked object only on the neighbourhood of the object.

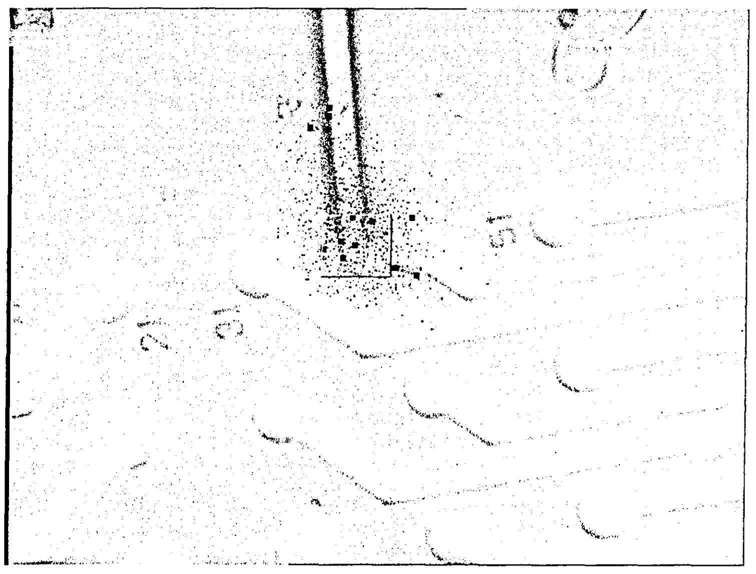


Figure 6.44: In spite of the blurred features of the pipette, its location is found. Big black square dots are the peak points.

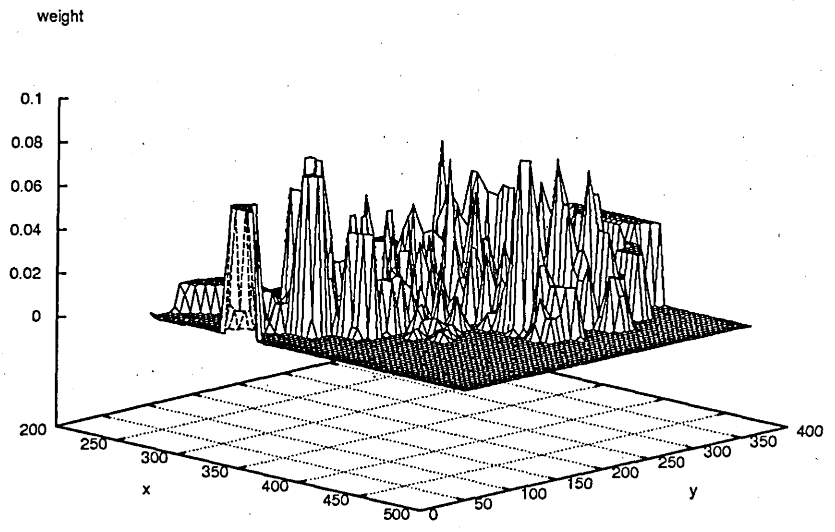


Figure 6.45: The heavy background clutter is illustrated by the existence of multiple peaks in the graph. In spite of the heavy clutter the pipette tip is well localised through further evaluation of the peak points.

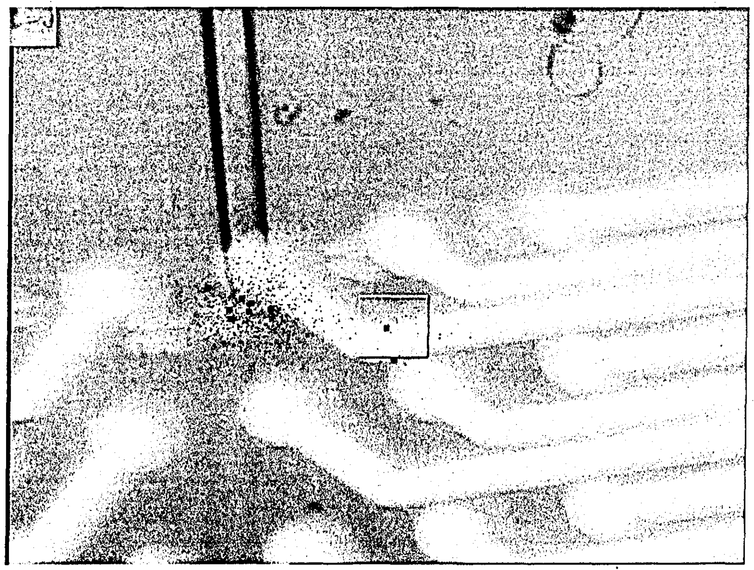


Figure 6.46: A rare case where the tracking has failed. The centre of the square is the tracked location.

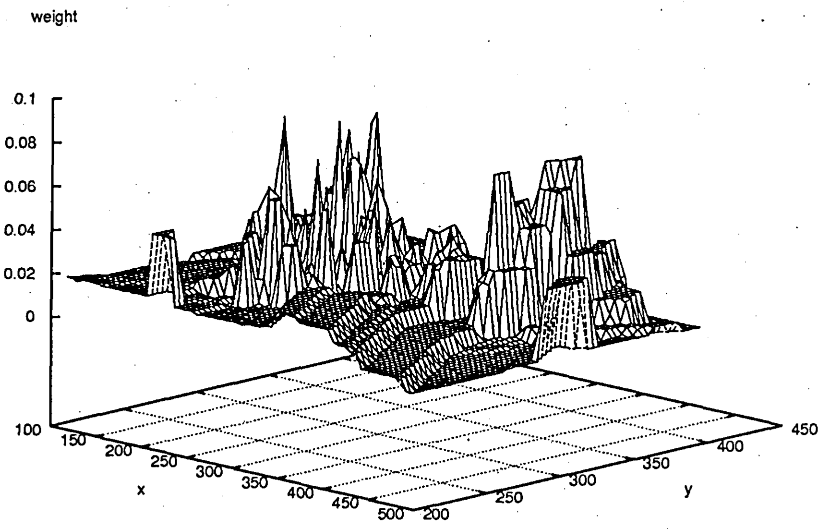


Figure 6.47: The graph illustrates that most of the peak point obtained, though the correlation of the edge template, are located around the pipette tip. But subsequent multivariate feature measure picked up the wrong peak point as the probable location.

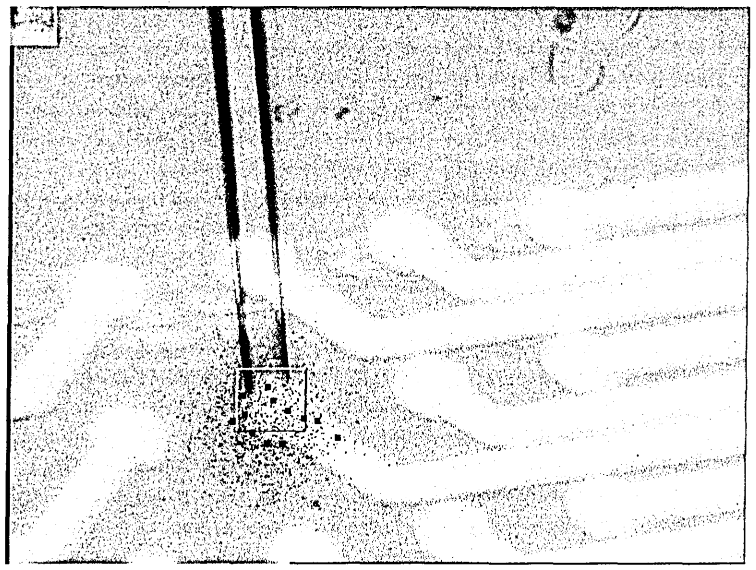


Figure 6.48: Another cluttered image, this one is due to the change of the background of the pipette and illumination of the scene. The small black square points are the peak points. Centre of the square is the tracked location.

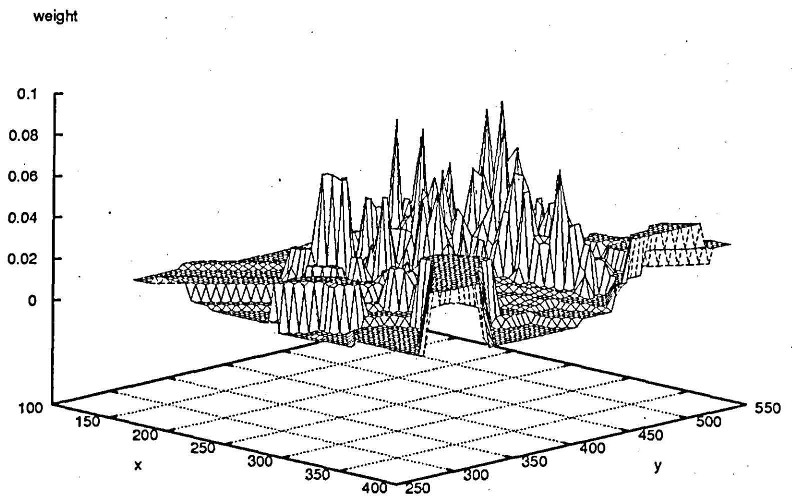


Figure 6.49: The graph illustrates the background clutter which gives rise to multiple peak points.

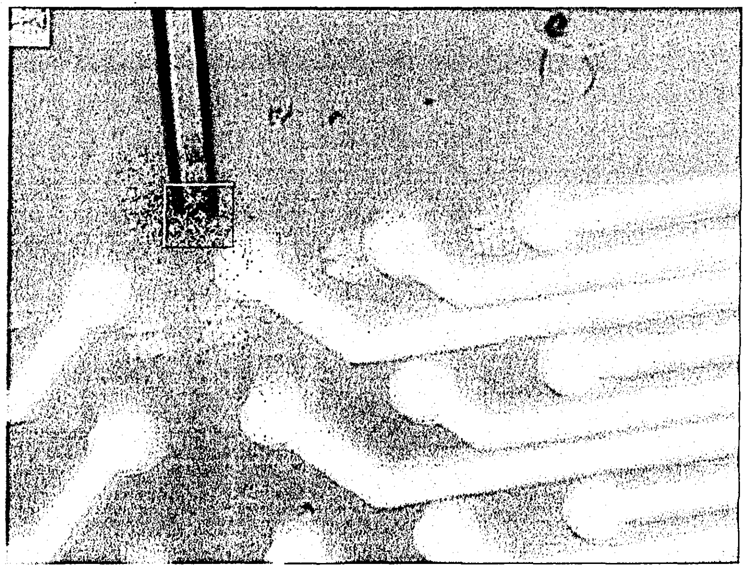


Figure 6.50: Last image of the tracking sequence. Magenta points are previous tracked locations.

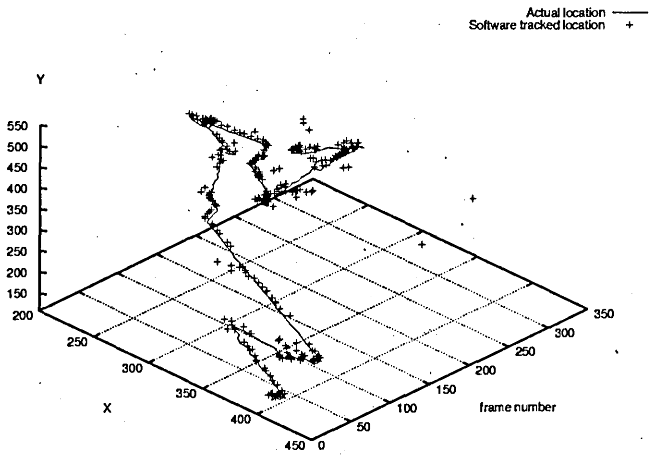


Figure 6.51: Actual location of pipette (line) and tracked location (crosses).

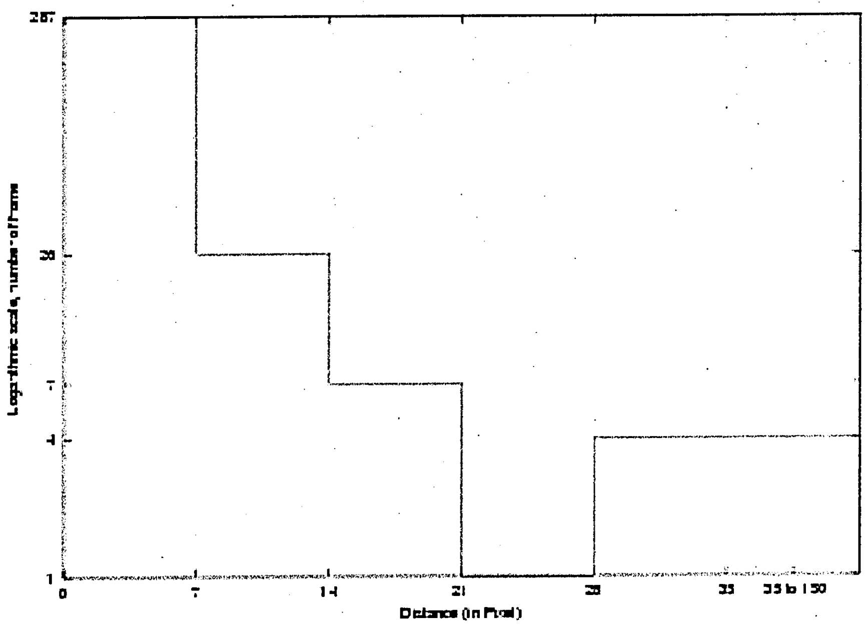


Figure 6.52: Number of tracked frames versus distance of actual location and tracked location.

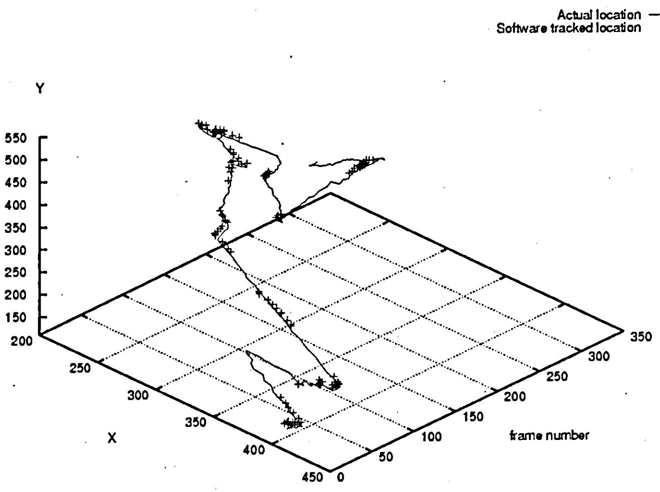


Figure 6.53: Actual location of pipette (line) and tracked location (crosses). Only tracked location of points with high probability is displayed

6.4 Testing the stencil Hough transform on the pipette video sequence

6.4.1 Experiments and results For comparison purposes with the particle filter, the algorithm presented in chapter 3, the stencil Hough transform was evaluated on the pipette video sequence. As a reminder, figure 6.54 presents 2 images extracted from the video sequence.

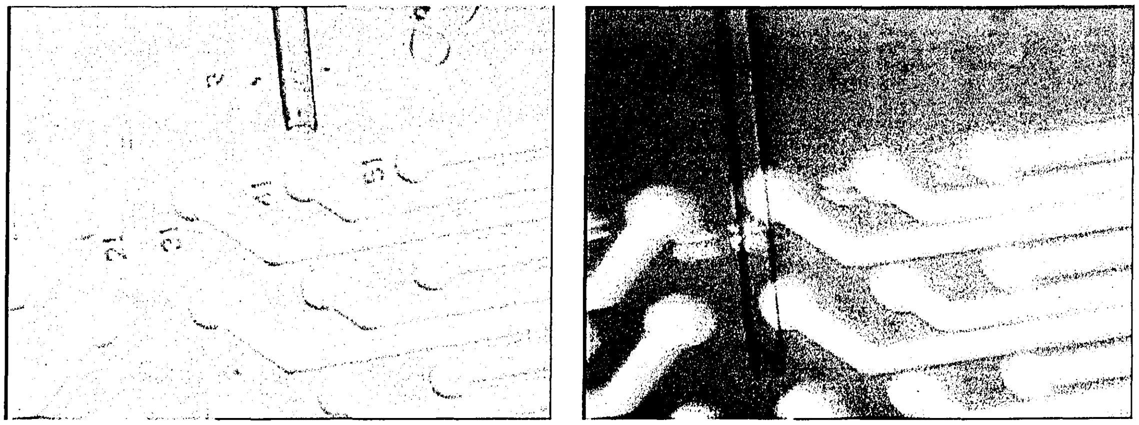


Figure 6.54: Two images from the pipette tip sequence. It is the same sequence that was used to discuss the particle filter algorithm.

A custom edge detector was used for the fast extraction of features. A Canny edge detector with 2 different settings was also tested, the feature extraction resulted to be 4 to 5 times slower but yielded more accurate tracking results. Our custom edge detector simply consists of thresholding the gradient of the image and, when multiple contiguous pixels with high gradient value are present on a same line, to select the pixel with the highest gradient value. The number of feature points is thus reduced and the filtered image consists of thin edges. Despite the simplicity of the method the edge extraction process is rather slow (around 200 ms per image). For real-time processing, images can be down-sampled and a multi-core CPU used.

Figures 6.55 to 6.59 allow us to visualise and compare the behaviour of the tracking algorithms for different settings of its components; table 6.4.1 provides numerical data that show the precision of the tracking. The best results were obtained with the first settings of the Canny feature detector and using the motion filter that is described later on. Although the motion filter visually improved the tracking in a

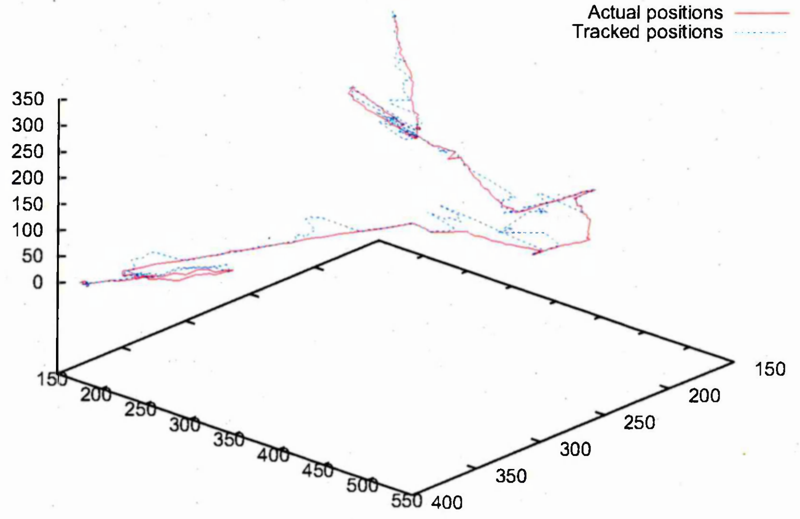


Figure 6.55: Comparison of the tracked positions and the manually determined positions of the pipette tip when using our custom edge detector.

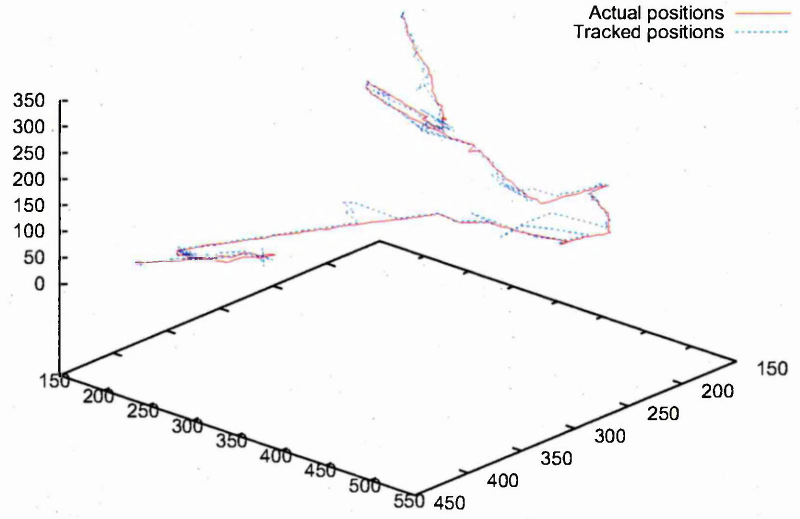


Figure 6.56: Comparison of the tracked positions and the manually determined positions of the pipette tip using the Canny edge detector with the first set of parameters.

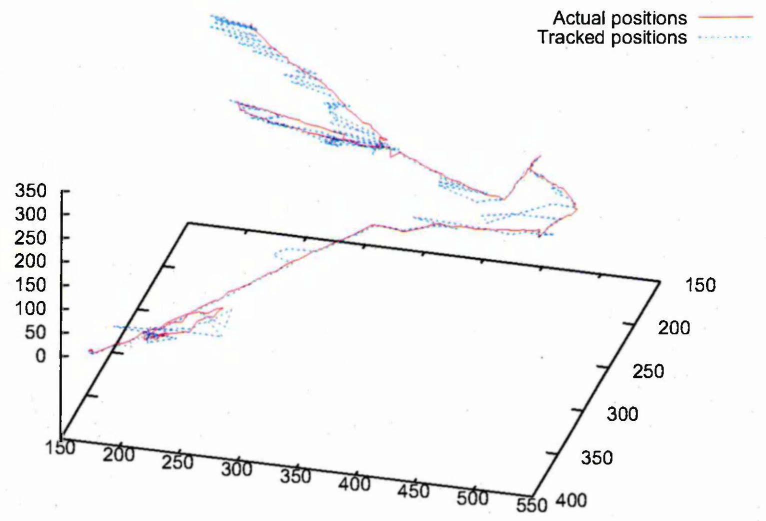


Figure 6.57: Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the second set of parameters.

noticeable way: when the tracking is observed the pipette tip appears to be tracked correctly even when none of its features appears in the filtered image, this improvement is not obvious and its utility might be questionable when looking at the table 6.4.1. Nevertheless the computational cost of this improvement is negligible and the increase of corrected tracked frames, although minimal, can make the difference between losing the track or not and, ultimately, minimising the number of costly re-initialisation steps.

6.4.2 The motion model The shape of the feature of the pipette, due to its transparency and changes in illumination, changes to an extent that for some frames most of the features of the pipette tip disappear and the stencil Hough transform can no longer locate the pipette tip. In the pipette sequence these changes span for a number of frames and the previous dynamics of the pipette can be used to estimate the tip position.

The following motion model was used: since the pipette tip has an erratic movement but generally tend to move in the same direction it was assumed that the

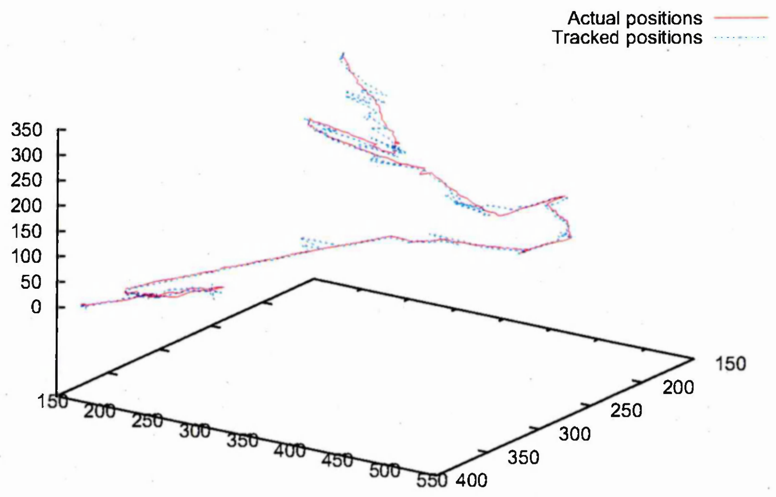


Figure 6.58: Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the first setting and the motion filter.

speed of the pipette is constant for a few frames and becomes null a few frames later. This allows the tip to be tracked for a few frames in case of occlusion while not getting too far off the pipette in case of a change of its direction; an event which probability increases as time passes. Of course if the pipette direction changes when it becomes occluded the pose estimation provided by the motion model would be incorrect. However, for the pipette sequence this does not occur. In general, if the cause of an occlusion can also be the cause of a directional change of the tracked object, motion models relying uniquely on the previous dynamic of the object would not be very useful.

$\hat{x}_t$ denotes the estimated position of the object after the evaluation of the measurements, $\hat{x}_t^-$ the position estimated using the motion model which is evaluated as follows: $\hat{x}_t^- = \hat{x}_{t-1} + v_{t-1}$ where v_{t-1} is the previous estimated speed of the object. All these variables are vectors, and for the pipette sequence they are 2 dimensional.

To take into account of the inherent uncertainty of the motion model which varies with the measurement an additional variable u_t is used. u_t represents the maximum distance the object can be from the current estimated position of the object at frame

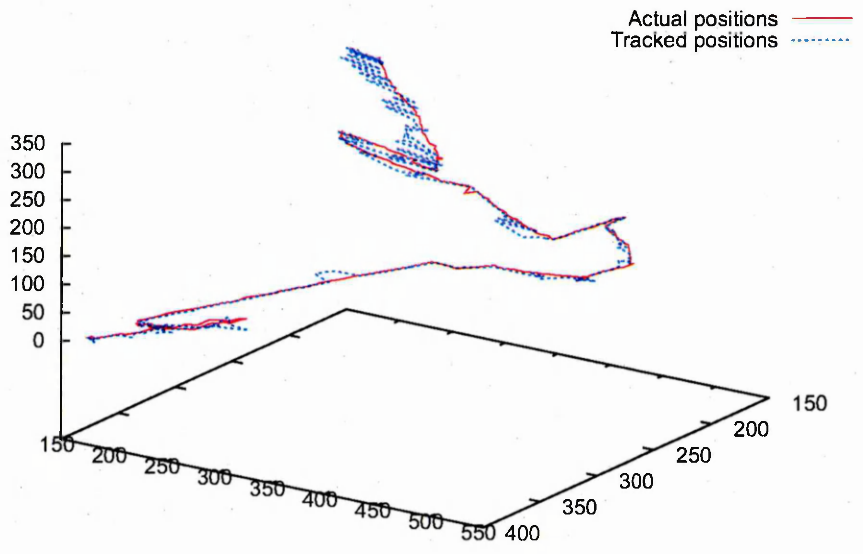


Figure 6.59: Comparison of the tracked positions and the manually determined positions of the pipette tip when using using the Canny edge detector with the second setting and the motion filter.

Edge detection method	Number of frames (out of 331) for which the distance between the tracked position and the manually determined position is:				
	< 6 pixels	< 12 px	> 12 px	> 20 px	> 40 px
Custom edge detector	176	220	104	76	49
Canny, first setting	224	262	65	38	12
Canny, second setting	206	220	77	54	20
With motion model:					
Canny, first setting	227	265	62	36	7
Canny, second setting	210	256	70	43	20

Figure 6.60: Tracking accuracy

Edge detection method	tracking speed (ms)			feature detection time (ms)		
	average	max	min	average	max	min
Custom edge detector	64	155	24	197	368	174
				feature collection time (ms)		
Canny, first setting	22	52	6	34	66	22
Canny, second setting	46	117	13	44	96	18

Figure 6.61: Tests carried out on an Intel Celeron Northwood 2.7 GHz CPU

t.

According to the validity of the measure given by the stencil Hough transform the parameters of the motion model (speed and uncertainty) are updated and the tracked location is determined. Algorithm 4 details the procedure.

Readers familiar with the Kalman filter would have noticed a number of similarities such as the recursive integration of the novel information about the speed of the object. There are also dissimilarities; because of the nature of the stencilled Hough transform, measurements are done exhaustively in a region of interest that is usually larger than the region of interest indicated by the motion model. The motion filter is only used as an auxiliary feature when the measurements are not satisfactory or to check that the result, that depends on the measurements, is logically compatible with the previous location of the object and its motion model. This is because the reliability of the measurement is believed to be considerably higher than the information that can be obtained from the motion model. Given the weak assumptions that could have been made on the motion model this is justified and results in this particular case of the Kalman filter where the motion estimation is used only when the image data does not allow to locate the position of the object. The motion filter can also be used to position the region of interest where measurement are made. However, given that the motion is small compared to the size of the region of interest, choosing the previous estimated position of the object as the centre of the region of interest was not altering the tracking.

Without the motion filter the size of the state space has to remain large to be able to recover the pose of the tracked object when it is occluded. By reducing the uncertainty, the motion model allows the size of the region of interest where to look for the object to be reduced, thus reducing the size of the state space along its translation dimensions. This also increases the speed of the feature extraction by reducing the region of interest in the image where to look for the object. For the pipette sequence, despite the weakness of the motion model,this reduction was not negligible: the size of the state space along the y-axis could be reduced by about 40%, while it remained unchanged along the x-axis.

6.4.3 Summary Compared with the particle filter algorithm the bounded Hough transform is faster. The particle filter was taking a few seconds per frame to track the object; this was essentially due to the matching method that consisted of cross correlating edge features in the images. Using a better measure with the particle filter would solve this issue but may add some complexity to the implementation; for instance, a contour model of the pipette tip could be used but for each new tracked object a new model should be used which adds another level of complexity for the users of the algorithm and requires an additional module to the tracking system. For rigid objects, real-time can be achieved comfortably if the motion is planar (rotation and translation) for more degrees of freedom except if the motion can be tightly bounded the particle filter with an efficient similarity method is likely to perform better.

The stencilled Hough transform did require significantly less tuning compared with the particle filter algorithm, the similarity method is very robust but less than what could be expected from the artificial data. This is because the pipette is self similar along its axis of symmetry which occasionally results in the pipette body to better fit the pipette tip stencils when the tip is occluded.

A major difference between these two techniques is that the stencilled Hough transform performs an exhaustive search on the region of interest while particle filters sample this region. Depending on the requirement of the application this may provide a theoretically more satisfying guarantee that the object is more precisely located.

Algorithm 4: A motion filter for the stencil Hough transform

```
Measurements are taken around $\hat{x}_t^-$ if *measurements give the position with a high value of certainty* then
    $\hat{x}_t$ is set to the best measured position
    $u_t = 0$
    if *the previous pose was known with accuracy* then
        $\hat{v}_t = \hat{x}_t - \hat{x}_{t-1}$
    else
        $\hat{v}_t = \frac{\hat{v}_{t-1} + K_1(\hat{x}_t - \hat{x}_{t-1})}{1 + K_1}$
if *Measurements give the position with an intermediate value of certainty* then
    $u_t = u_{t-1} + U_1$
    for *The n poses having the best value in decreasing order of likelihood* do
        if *The pose is compatible with the motion model* then
            $\hat{x}_t$ is set to this pose
            if *the previous pose was known with accuracy* then
                $\hat{v}_t = \frac{\hat{v}_{t-1} + K_2(\hat{x}_t - \hat{x}_{t-1})}{1 + K_2}$
                $u_t = U_2$
                break
            else
                $\hat{v}_t = \frac{\hat{v}_{t-1} + K_3(\hat{x}_t - \hat{x}_{t-1})}{1 + K_3}$
                $u_t = U_2$
                break
    if *No likely pose is compatible with the motion model* then
        $\hat{x}_t = \hat{x}_{t-1} + \hat{v}_{t-1}$
        if *Motion filter parameters have been updated in the last few frames* then
            $\hat{v}_t = \hat{v}_{t-1}$
        else $\hat{v}_t = 0$
if *Measurement do not give any satisfactory value* then
    $u_t = u_{t-1} + U_1$
    $\hat{x}_t = \hat{x}_t^- = \hat{x}_{t-1} + \hat{v}_{t-1}$
    if *Motion filter parameters have been updated in the last few frames* then
        $\hat{v}_t = \hat{v}_{t-1}$
    else $\hat{v}_t = 0$
```

Where K_1, K_2 and K_3 are scalars that can depend on the measurement value and the uncertainty of the motion filter; and U_1 and U_2 are also scalars that can represent the maximum speed of the object and a distance that can depend on the value of the measurement.

Chapter 7

Conclusion

7.1 Contributions

An original framework that analyses a shape relative to its state space has been established. This results in an additional step that can be performed offline and that refines the shape representation of a known rigid object. A set of characteristic points that are relevant for the robust identification of an object is thus determined, which increases the performance of tracking, recognition and detection algorithms.

For a given state space, it has been shown how a variation of the Hough transform can be utilised to determine whether a set of points uniquely characterises an object state. This algorithm works for sets of points, extracted from a rigid object shape, when the object motion can be bounded.

The concept of the stencil estimator has been introduced and it has been shown how it is useful in improving the performance of the bounded Hough transform in terms of computational speed and memory space requirements. The modified tracking algorithm, which is independent of the selected feature detector, is referred to as the stencilled Hough transform. Performance comparisons have established that the approach is well founded. It has also been demonstrated that the stencilled Hough transform can be parallelised efficiently with minor modifications.

The particle filter was presented by describing a simple example of its usage, it is hoped that this will aid in obtaining a deeper understanding of this tracking algorithm. Some generic improvements have also been proposed: the clustering of particles, the over-weighting of the particles, partial reinitialisation and the introdution of additional stages to incorporate other measurements in order to enhance the sampling of the new data. Some of these improvements, such as partial reinitialisation and the usage of additional measurements, had been already proposed previously by other researchers, but in a slightly different manner.

Additionally, as presented in section 6.1, it has been shown how the bounded Hough transform can be adapted for tracking, with 4 degrees of freedom (x-y translations, rotation and depth translation), of rigid objects under a microscope. The evaluation of the depth translation was performed using a stack of object images taken at different depth levels. This takes advantage of the fact that due to the narrow depth of field of microscopes the appearance of an object changes with its distance to the microscope lens.

7.2 Future research

It has been argued [15][99] that high curvature points convey significant information about a shape and as a consequence characterise shapes well. It would be useful to investigate if the selection of these points for a given shape also characterises the shape with the definition proposed in this thesis.

Three main directions to extend the work presented in this thesis are suggested:

1. Increasing the search space to obtain a detection or recognition algorithm: in this thesis, since the focus was on tracking, the transformation space or state space was reduced to encompass a region of interest where the object was previously located. Further tests need to be performed to determine if it is possible to detect a shape in a whole image by simply increasing the area of interest and to correlate the evolution of the memory usage when the number of dimensions of the state space increases. If we consider the stencilled Hough transform algorithm, the main issues are currently the quantity of memory available and the number of features extracted. At the end of chapter 4 we suggested an algorithm (algorithm 1) for object detection. This algorithm may be implemented in many different ways and the stencilled Hough transform algorithm is an efficient way to implement stages three and four when the search space is relatively small.

2. Enhancing the object representation by selecting more discriminative feature detectors: another way to increase the robustness of the stencil estimator

is to incoporate additional information such as the gradient direction of the features or its colour, reducing the necessary number of feature points needed to characterise the object state. If distinguishability can be quantified, for instance features may belong to one of n categories (*e.g.* 8 directions for the gradient, 12 hues for the dominant colour of the feature surrounding patch *etc.*) it may be possible to further develop the framework described in this thesis to determine which feature detector, in terms of its speed and distinguishability of its feature, can be used to optimise the speed of the stencilled Hough transform.

3. Generalising the algorithm to track objects in 3-D: the bounded Hough transform has been shown to work with 3-D objects for movements in space. The stencilled Hough transform can be adapted for projective transformations of a 3-D object representation. One possibility is to use a stack of images of different object views.

The template reduction scheme, originally developed for the stencil algorithm, still needs to be tested with different shape matching algorithms to evaluate its robustness and the performance enhancement it provides.

Appendix A

Fitting a square to a set of points

A.1 3 points case

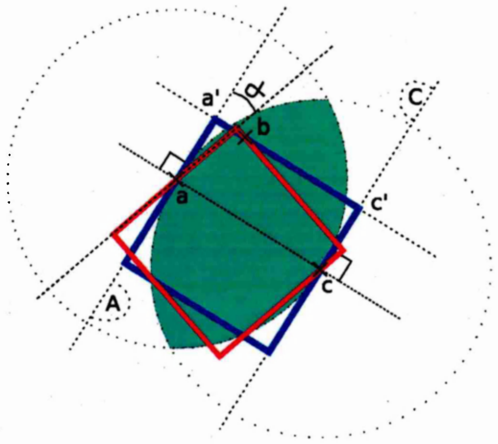


Figure A.1: An infinite number of squares can fit 3 points.

We shall use the following notations: (ab) for the line passing through the points a and b, $[ab]$ represents the segment having vertexes a and b and $|ab|$ the distance between a and b.

It is trivial that an infinite number of squares can fit 3 aligned points.

Consider 3 unaligned points as shown in figure A.1. The 2 points furthest apart are first considered. In the case of figure A.1 these 2 points are a and c. The third point belongs by hypothesis to the intersection of the 2 discs of radius $|ac|$ and with the centre these 2 points. It is then possible to construct a square intersecting these 3 points by considering that a and c belong to two opposite edges perpendicular to (ac), respectively called A and C . One of the edge of the square goes through the last remaining point, in our example b. This is possible because if b is projected on line A or C on respectively a' and c', then $|aa'|$ or $|cc'|$ are smaller than $|ac|$. The blue square of figure A.1 has thus been obtained.

It is in fact possible to construct an infinite number of squares that passes through these 3 points. Instead of taking the edges passing through a and c perpendicular to (ac), one can consider them to have a slightly different angle. We call this angle α. A square fitting the 3 points can be constructed this way as long as b remains between these 2 lines and that its projections on the 2 lines a' and c' are such that $|aa'|$ and $|cc'|$ are smaller than the square edge size. The square edge size is $|ac|\cos(\alpha)$. When $\alpha = 0$ the maximum value for $|aa'|$ and $|cc'|$ is $|ac|\sin(\arccos(0.5))$, this would happen if c was at the intersection of the circles. This gives a comfortable margin for α to vary around 0 and still satisfy the previous above mentioned conditions that allow a square to fit the given points. The red square (or clear grey square) of figure A.1 is an example of such a square.

A.2 4 points case

Consider that any 3 points of the configuration are not aligned. For this case we could not determine the conditions that ensure that more than one square fitting four points can be constructed. As shown in chapter 4, figure 4.4, some configurations of 4 points can be fitted by only one square, nevertheless note that the presented configuration has 3 points aligned. However we present a few starting ideas to explore the problem.

A necessary condition for four points to belong to square edges is that it is possible to make a convex quadrilateral that has these four points as its vertexes. Considering the 3 points a, b and c of figure A.2 to have such a condition, a fourth point has to be taken on the non hatched area and outside the triangle abc. d and e are examples of such points. This may not however be a sufficient condition.

We now examine what happens when trying to generalise the constructive method presented for the 3 points case. Figure A.3 exhibits an example where 2 points are furthest apart. Like in the previous method we assume that these points belong to opposite edges of the square. This leads to 2 parallel lines going through the points as shown in figure A.3.

If we assume that the 2 remaining points belong to the same edge there is at

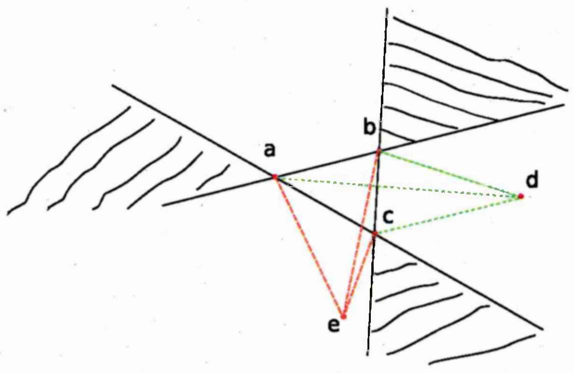


Figure A.2: *abcd* and *abce* are 2 four point convex configurations.

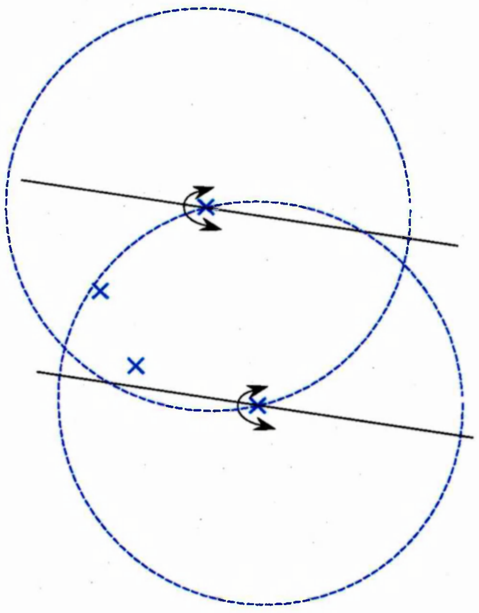


Figure A.3: For this configuration of four points we consider the 2 points furthest apart.

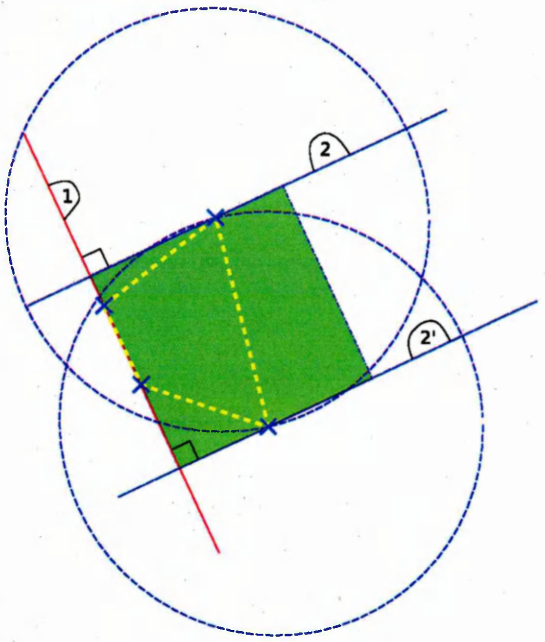


Figure A.4: The green shaded square was constructed by assuming that the 2 points furthest apart belong to opposite edges and the 2 remaining points to one of the edge.

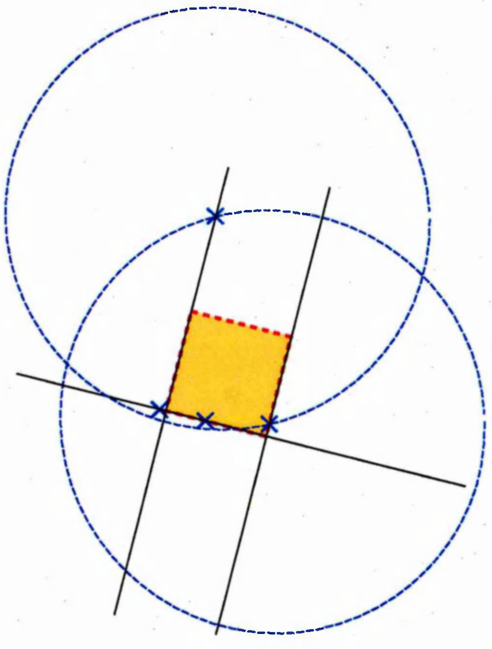


Figure A.5: It is not always possible to fit a square that intersects the 4 points using the assumption that the 2 points furthest apart belong to opposite edges and the 2 remaining points to one of the edge as shown with this configuration of four points.

most one square satisfying these conditions. Such a square is shown on figure A.4. However it is not always possible to fit such a square as the example of figure A.5 demonstrates this.

A special case is considered in figure A.4. Indeed, the convex quadrilateral which vertexes are the four points outlined in the yellow dotted line is such that the longest distance between the four points is an edge of this quadrilateral. If the longest distance belongs to a diagonal of the quadrilateral then the assumption that the 2 points that are furthest apart belong to opposite edges of a square is inconsistent with the assumption that the 2 remaining points belong to the same edge of a square.

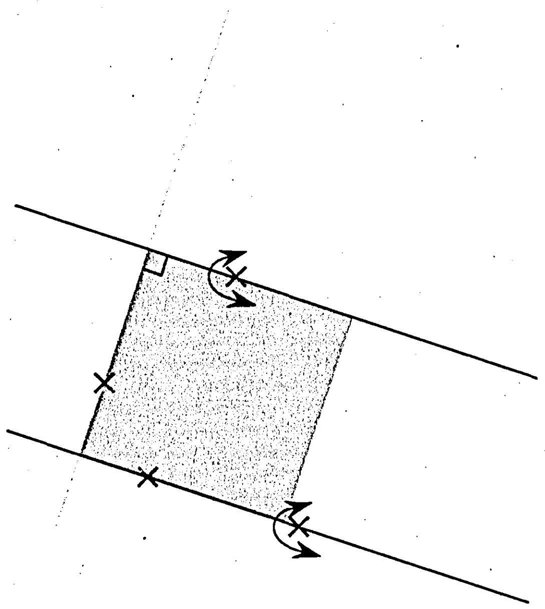


Figure A.6: If the longest distance is an edge of the convex quadrilateral and we assume that the two remaining points belong to different edge of the square there exists at most two squares that fit the four points.

We can also assume that the two remaining points belong to two different edges of the square. If once again, we consider the condition that the points furthest apart belong to the edge of the convex quadrilateral; to construct such a square the parallel lines can be rotated until one of the two remaining points belongs to one of the lines and the other point is between them. This might not be possible as shown in figure A.7 or possible as shown in figure A.6

So, examining the problem by considering the 2 points furthest apart, we see that multiple cases are possible and we have not reached a better conclusion than some 4 point configurations can be fitted by one or more squares and some cannot.

If one is inclined to solve this problem a few ideas are suggested hereafter. A different approach to analysing the problem would be to divide the points into 3 groups according to the 6 intersection points of the lines of the convex quadrilateral which vertexes are the four points. By assuming that parallel lines intersect at infinity, parallel lines could be taken as a separate case or not, according to the need of the analysis. Figure A.8 illustrates this idea. Yet another approach would be to establish a system of equations that need to be satisfied for constructing a square and to solve it when possible. Figure A.9 illustrates such an approach.

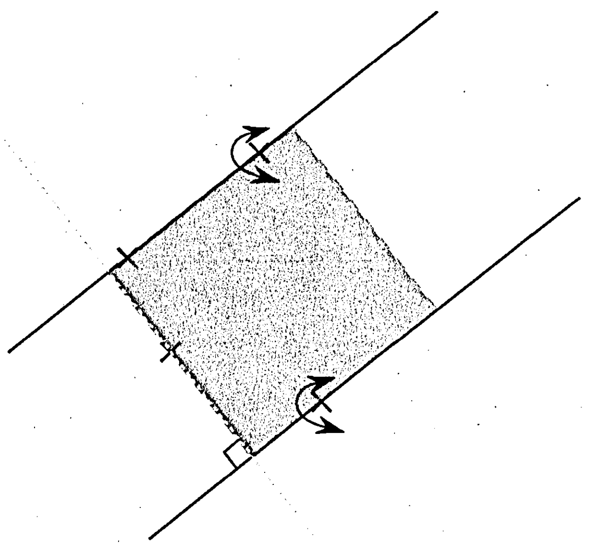


Figure A.7: An example when the four points are fitted by a square assuming that the two points furthest apart are on opposite edges of the square and the remaining points belong to different edges.

A.3 5 points

We consider that any 3 points of the configuration are not aligned. Since the square has four edges, at least 2 points belong to a given edge. 2 points are chosen such that a line passing through them has the 3 remaining points on one of its side. If a convex pentagon can be made by joining the points with lines then there are 5 such possible choices. For each of these choices we are going to evaluate the maximum number of possible squares passing through all 5 points. On the remainder of the section we call A the edge of the square passing through the chosen pair of points.

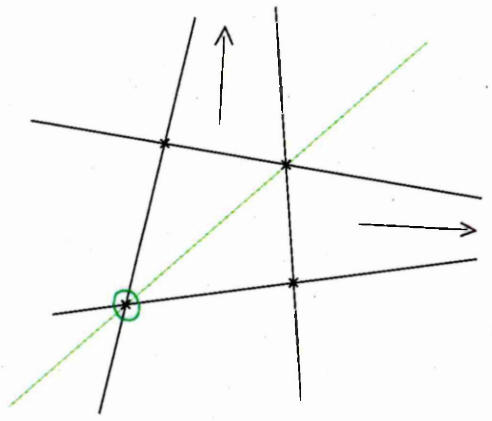


Figure A.8: By considering the direction of intersection, represented by arrows, of the plain black lines the four points can be separated into 3 groups: the point circled in green, the other point on the dashed green line and the two remaining points.

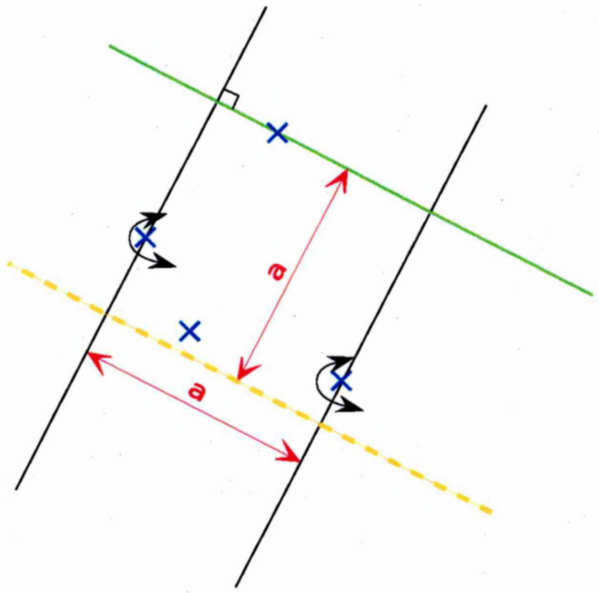


Figure A.9: 2 Points are taken randomly, by assuming that they belong to opposite edges, for instance, a system of equations can be written and solved to check if it is possible to construct squares that are intersecting this four points. Curved arrows represent the possible rotations of the parallel lines.

The 3 points are not aligned so either (1) the three points belong to 3 different edges, or (2) 2 points belong to one edge and the last remaining points to a different edge.

1. In this case at most one square can pass through all 5 points. We consider the 3 lines perpendicular to A that passes through the 3 remaining points. If there are only 2 lines it is categorised as case (2). No demonstration is given but the point belonging to the middle line must belong to the opposite edge of A and the points on each side to the adjacent edges of A. Given the points defining the direction of A and the opposite edge of A and considering the previous constraint only one square is compatible with these points. For the configuration to be compatible with a square shape, the 2 remaining points must belong to this square.

2. A minimum requirement for the configuration of points to be compatible with a square shape is that 2 of the 3 remaining points belong either to a perpendicular line or to a line parallel to A. Without considering any other constraints at most 6 squares can be compatible with this point configuration.

As a conclusion, if we consider a configuration of 5 points, only a small finite number of squares can pass through this configuration. Moreover if the points were taken randomly we believe that it is highly probable that there may not exist a square that passes through all points. Indeed in case (1) the 2 last points must belong to 2 segments of the plane for the configuration to be compatible with a square. This is unlikely even if we consider a small margin of error and a bounded surface for the possible position of the points. For case (2) to occur, 2 points out of 3 must belong to a perpendicular line or a parallel line to A which is also unlikely. Again considering a small margin of error and a bounded space the author believes that it is possible to prove that this is unlikely.

A.4 Perspective transformation

If a perspective transformation is considered, one of the unique invariants that remain for geometric shapes is the alignment of points. Thus, when undergoing a perspective transformation a square can become a quadrilateral. With 8 points, any 3 of them not aligned, in a convex configuration it is possible to construct 2 quadrilaterals that fit the points. This can be seen in figures A.10. Therefore to determine a transformation that a square undergoes to fit points a minimum of 9 points, out of which there is necessarily at least one group of 3 points aligned, is required. This example is given to underline the fact that the number of dimensions of transformation space under consideration affects the minimum number of points required to completely determine an object location.

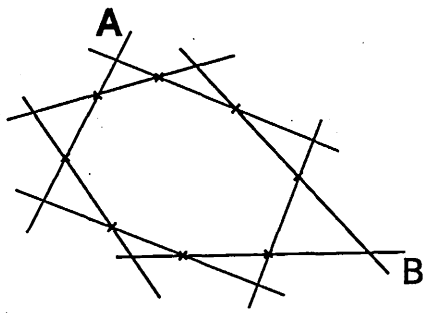


Figure A.10: 8 points, any 3 of them not aligned, in a convex configuration can be fitted by 2 quadrilaterals.

Appendix B

Dense disparity map using epipolar geometry

B.1 Depth maps of 3-D scenes

Having a 3-D models of the objects that are being tracked is necessary to be able to track them in full 3-D. However, obtaining a 3-D model of an object is not straightworward. There currently exists a number of commercial applications [100]. To achieve this goal, in the following sections we shall explore the well known technique of stereo vision. Knowledge about the geometry of a scene provides interesting clues to identify and estimate the pose of an object.

We will conclude that stereo vision, although it has some interesting application domains, is, in our opinion, and despite the effort of many researchers, not appropriate to capture the 3-D geometry of an object. This is because better, simpler and more efficient techniques are available.

B.1.1 Stereo vision Stereo vision is also known as stereopsis, stereoscopic vision or binocular vision. These techniques aim to provide a depth map of a scene using two cameras. It combines two images taken from two slightly different points of view.

This process can be compared with the human vision system. When our eyes view a scene, the images drawn on each retina differ by a small degree. This is illustrated by figure B.1 [101]. From these two corresponding images and the position of the eyes, the brain creates a perception of depth.

The whole stereopsis process can be divided into the following three stages:

1. Camera calibration
2. Image correspondence
3. Triangulation

The method has been generalised to more than two cameras, *i.e.* trinocular and beyond [102].

B.1.2 Camera calibration The mapping of the world - the geometry of which can be modelled with a 3-D vector space - to a 2-D discrete space, the image, has already been modelled successfully through various methods [42]. According to the chosen model, a slightly different set of parameters have to be found. These parameters depends on the characteristics of each camera. The process of determining these parameters is known as camera calibration.

Camera calibration is still an active field of research, although the technology is mature enough so that different implementations are freely available. Such implementations can be found for instance in the Mimas, OpenCV and Gandalf libraries. Thus, the camera calibration problem boils down to identifying a suitable model for our requirements and an implementation to obtain the model's parameters.

Camera parameters are generally classified into two classes: the intrinsic and the extrinsic parameters of the camera. The intrinsic parameters are the focal length, the coordinates of the principal point and a few parameters to model the geometric distortions characteristic of the lens system. The extrinsic parameters are the position of the camera, *i.e.* location and orientation compared to an arbitrary external frame.

The process of calibration needs to be performed only once and thus it can be conducted off-line. It involves taking images of a scene where 3-D points of the scene are known. By finding the correspondence points in the resulting images, the parameters are found by solving a system of equations. Once the camera is calibrated it is possible to associate a 3-D ray to each pixel of the image as illustrated in figure B.2 or to predict the 2-D location in an image of a 3-D point of the scene.

We have tested a method that uses a calibration object [42]. This object is a grid similar to a chess board with known measures. Figure B.5 shows the calibration grid.

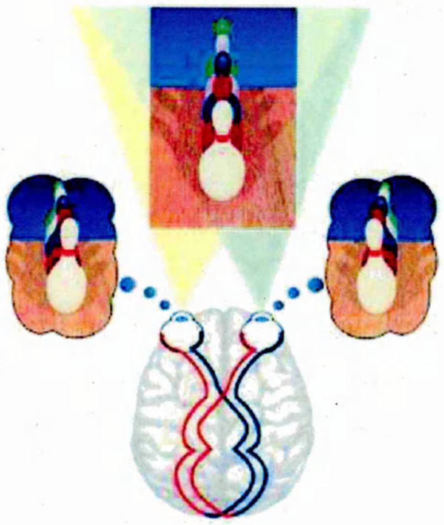


Figure B.1: The images seen by each eye are slightly different. *Image from the Optometrists Network website.*

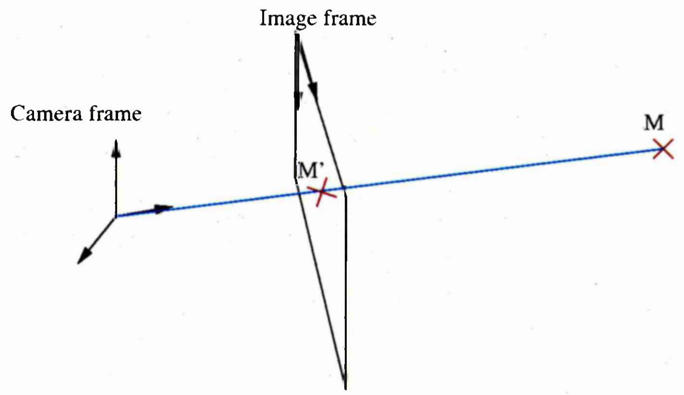


Figure B.2: By obtaining the intrinsic parameter it is possible to determine where each point expressed in the camera frame will be projected onto the image.

Once the intrinsic parameters of the camera are known it is possible, for instance, to project a model of an object in the image as illustrated in Figure B.3. Using the intrinsic parameters of the camera we have mapped a cuboid having the dimensions of the chess board with the chess board in the image. The mapping was performed manually by trials to estimate the location and rotation of the chess board.

This process can be automated if there exists a way to estimate the pose of the object. This is actually the basis of model-based 3-D pose estimation which was the initial direction of this research on stereo-vision. The results of some experiments made with a Rubik's cube are presented in section C.2 of this thesis.

In figure B.3, the white line at the top of the image is curved due to lens deformation. It is a line we have mapped on the image to the approximated, manually estimated, position of an edge of the power supply unit at the top left corner of the image. It demonstrates that the deformations of the image by the lens system are taken into account by the model when a model is re-projected onto the image.

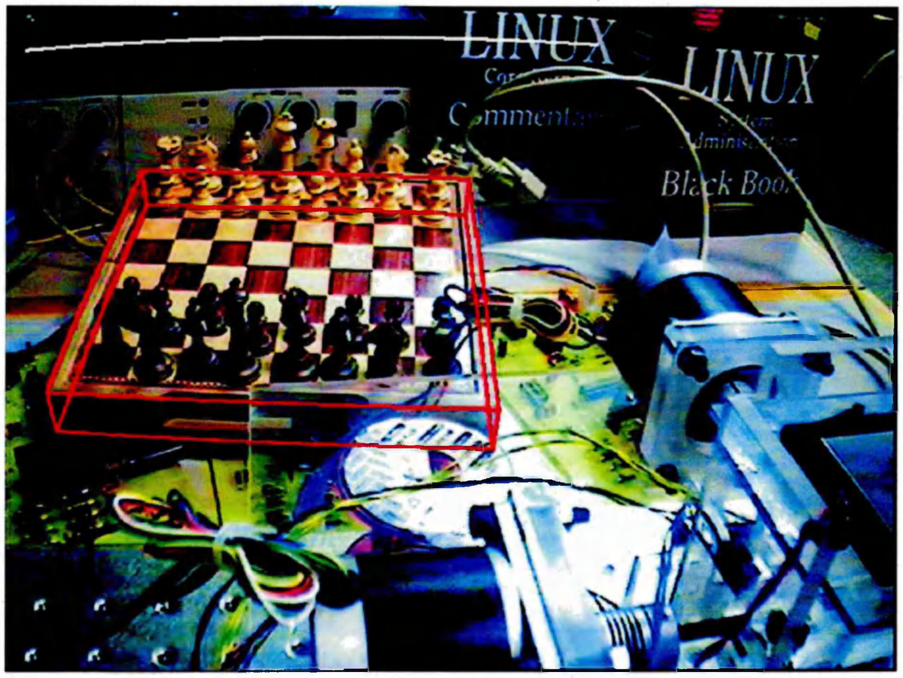


Figure B.3: A red parallelepiped and a white line projected on the image using the image formation model

Distortion parameters can also be used to rectify an image. A rectified image is an image such that lines onto the image are projections of lines from the real world. This is illustrated by figure B.4.

B.1.3 Stereo calibration When a scene is captured from multiple points of view, knowing the relative location of the cameras is a preliminary step towards determining the geometry of the scene. In the case of a stereo rig the process of determining the relative position of the two cameras is known as stereo calibration.

Once calibration is achieved, it is then possible to obtain standardised images. Standardised images have been *rectified* such that when a point can be seen on the two images, this point belongs to the same base line on both images. Once the stereo-calibration has been performed off-line, the image standardisation process is reduced to a warping of the image pair. Image standardisation is useful as they provide a way to reduce the dimensionality of the correspondence problem.

Figure B.5 shows the two original images acquired with the stereo rig and the standardised images.

B.1.4 Correspondence problem In its more general formulation, the correspondence problem consists of finding, for a point in a scene, where it appears (if it appears) on different images taken from different points of view.

In the special case of stereo vision since the two points of view are close, most of the points are visible in both images. Moreover, the two images can be transformed in such a way that two corresponding points would appear on the same line. After such a transformation the images are said to be *standardised.*

A *disparity map* can be created to represent the relative position of two corresponding points. Such a map can be visualised as an image as shown in figure B.6. Dark pixels represent small disparities and thus far elements and brighter pixels

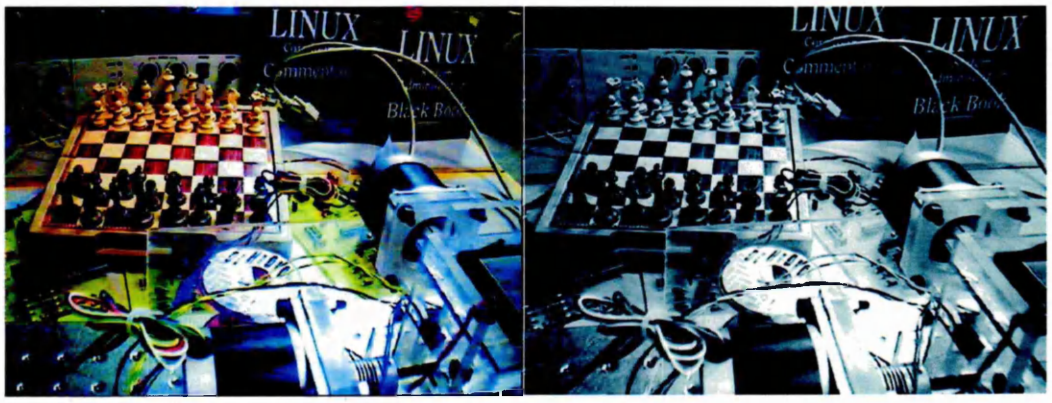


Figure B.4: The right image is the rectified left image. Notice that the power supply edge highlighted in previous figure appears straight.

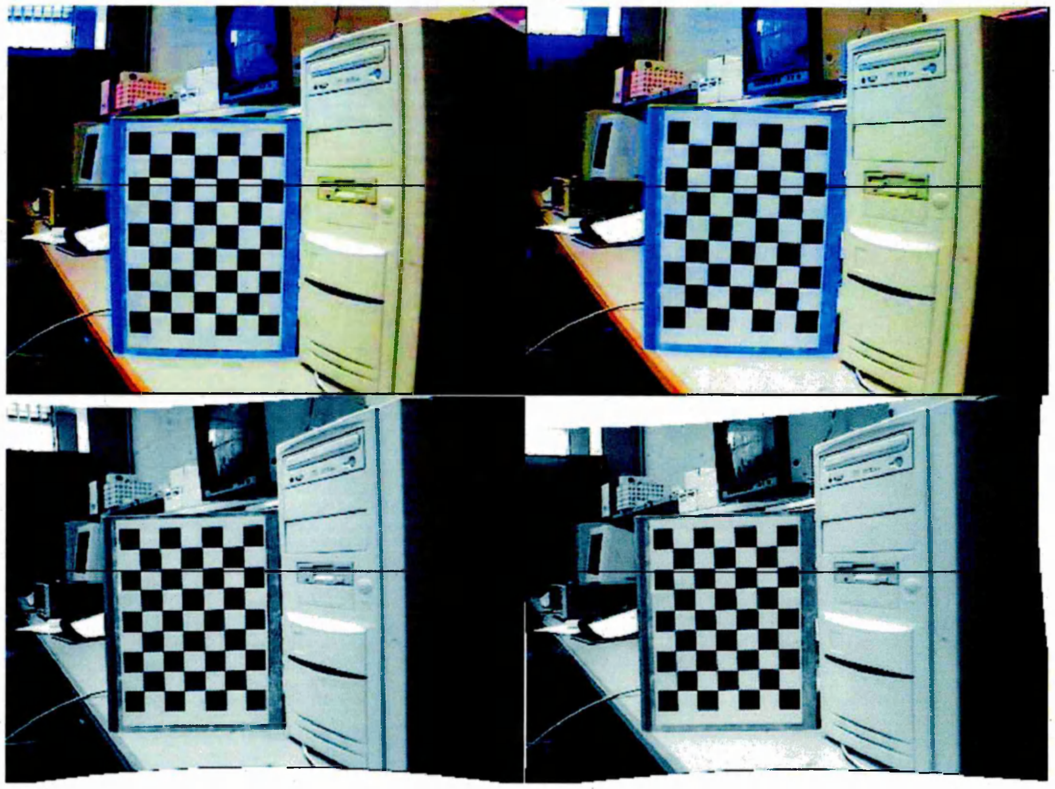


Figure B.5: Top: pair of images taken by our stereo rig. Bottom: the same pair after standardisation. Black lines have been drawn to exhibit the alignment of the corresponding image elements.

represent larger disparities and thus nearer elements.

Methods to solve the correspondence problem can be divided into two categories: sparse disparity maps where only the most reliable features are registered and dense disparity maps where a corresponding point is attributed to almost every points of the image. The usage of constraints such as continuity are used to match untextured surfaces. The second formulation is ill-posed as some points do not have correspondent points on the other image. However in the case of stereo images these points are a minority. Dense disparity maps are less reliable than sparse maps, however they give an approximation of the distance between the camera system and the object for almost every pixel.

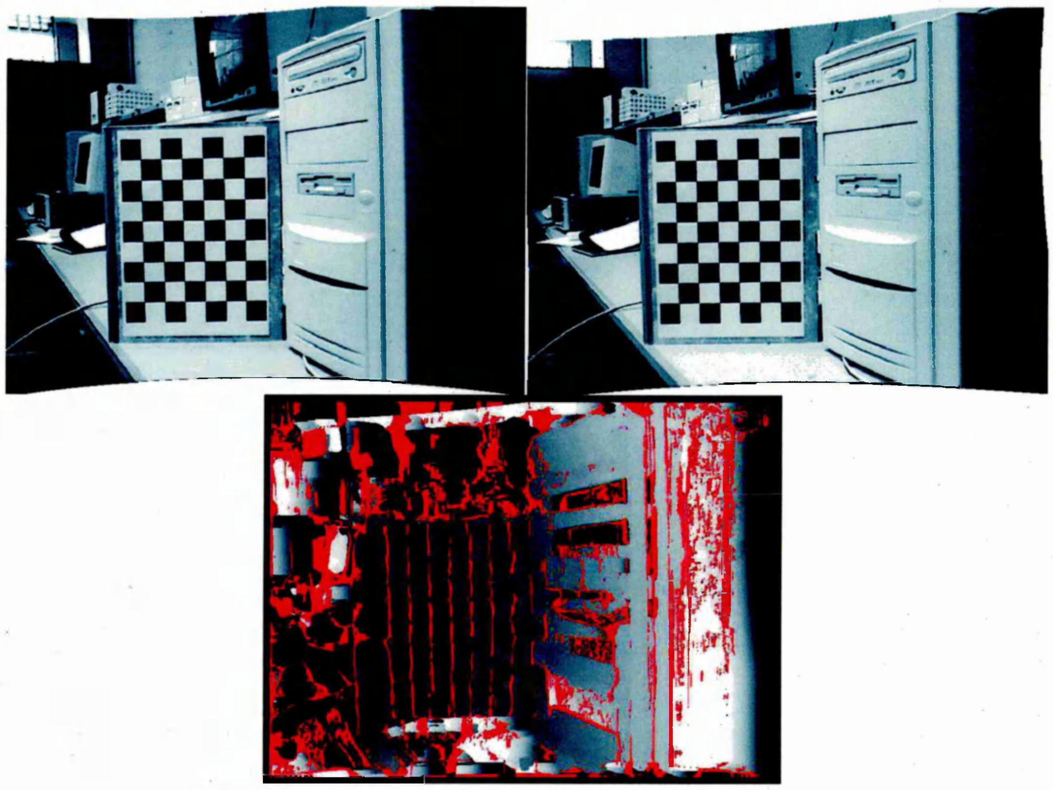


Figure B.6: Pair of standardised image and their disparity map. Red points are displayed when uncertainty is too high.

B.1.5 Discussion The major issue of dense disparity map evaluation is computational complexity. To illustrate this, assume that there exists a reliable method to decide if two pixels from different images correspond to the same point. If a brute force comparison algorithm is used to compare each pixel of one image with each pixel of the other image this hypothetical method would have to be used n^2 times where n is the number of pixels. So for relatively small images, say 320×240 pixels, the method would have to be applied around 6×10^9 times.

Considering that the hypothetical comparison method may require a non negligible amount of time, it appears that obtaining dense disparity maps, in real-time, on an entry level desktop computer is computationally expensive. As a result, many techniques have been developed that tries to limit computation. At the time of writing, a cost/energy minimisation formulation of the problem solved using a max flow/min cut algorithm [103] is one such approach [104][105]. At the point of implementation we thought that this technique was too slow for our purposes. However, Boykov and Kolmogorov [106] have shown that it was possible to implement graph cuts for real-time applications. In a 2007 seminar, Blake [107] mentioned the existence of a graph cut implementation, for standard PC, operating at a rate of 2 million pixels per second. It should be noted that graph cut algorithms provide an exact solution for a certain class of problem [108] modelled by a random Markov Field. They have also many other applications, notably for image segmentation [109][110], image reconstruction and more atypical image manipulations such as the creation of digital tapestries [111].

Another classic approach consists of using standardised images as described above. Thus, the search of a corresponding point is reduced to a line, transforming a 2-D problem into a one dimensional one. To determine if two pixels correspond to the same physical entity, measures, based on intensity or colour of the pixels, are used. To overcome major problems such as illumination changes, noise, drift between lines and the lack of texture, a window of surrounding pixels is taken into account. Further discussion on this approach is provided in appendix B.

For reference purposes other approaches have been tried [112]: simulated annealing, cooperative algorithms, dynamic programming, divide and conquer algorithms to cite just a few. The implementations are often more complex and do not provide significant improvement on the quality of the result or speed of the algorithm.

B.1.6 Summary Although stereo-vision has naturally emerged for biological "systems" and proved to be useful in various applications [104], such as segmentation or in robotics for obstacle avoidance, for systems aiming at capturing the geometry of a relatively small object it is impaired with:

1. unreliability when the object has insufficient texture to correctly evaluate the depth map.

2. imprecision since depth accuracy is decreasing proportionally to the inverse of the distance between the camera system and the entity position.

3. slowness due to computationally expensive techniques to evaluate the depth map. Although this is not an issue anymore due to graph cut techniques.

Capturing a model of an object is better tackled using structured light, like a laser or a striped projector, or time of flight [113] methods. Therefore, stereo vision has to be considered with circumspection for model building of objects since other technology can achieve the same purpose by evaluating the 3-D geometry of a visible scene in a more accurate, efficient and precise manner.

B.2 Model building alternatives

Another approach that can be used to locate a 3-D object is to determine object features that are highly identifiable despite perspective changes, or more simply, affine deformations. In his thesis [114] Johnson introduced the concept of spin images which has this property. Another kind of feature that is robust to affine transformations is referred to as local image descriptors. Amongst the most reliable such features are scale invariant feature transform (SIFT) [115][116] features and gradient location and orientation histogram (GLOH) features [117]. Mikolajczyk and Schmid [117] have done remarkable work in implementing a wide variety of popular and efficient local feature descriptors to compare them. Principal component analysis SIFT (PCA-SIFT) [118] and Speeded up Robust Features (SURF) [119] are two other interesting methods for local image invariants. In [120] Mikolajczyk *et al.* evaluate combinations of five region detectors and five region descriptors for the recognition of object classes. Results indicate that the usage of the Hessian-Laplace region detector combined with the extended SIFT (GLOH) image descriptor performs best for object class recognition. For an overview of image recognition videos of Pietro Perona conferences are available online [121] [122].

Having a 3-D model with these kind of features, it would then be possible to determine the pose of an object using a random sample consensus (RANSAC) algorithm [123].

In order to acquire the geometry of a 3-D model a laser line can be used. Faucher [124] and Boissenin started to implement such a solution, to acquire the 3-D of the object it is positioned on a rotational platform. The project, named Bright, is available on the web. It makes use of the Mimas library.

Structure from motion (SFM) determines an object geometry by analysing objects movements relatively to a camera. Therefore no laser line is necessary. Chang and Wong [125] proposed to solve the SFM problem by using a two-stage bundle adjustment [126] method. The first stage uses an approximated model to estimate the pose of the object. The second stage uses the pose information to refine the model structure. The two stages are executed repeatedly until the difference between the observed data and data re-projected from the estimated model is minimised. The tracking of the object is done using Lowe's method [127]. A method is used to track and estimate the 3-D pose of an object using a 3-D model of the object. A full perspective model of the camera can be adopted. A Newton minimisation method is used to solve the set of equation [128][129].

Wong *et al.* [130] discussed a method to acquire 3-D object using a video camera for the purpose of recognition and pose estimation.

B.3 Implementation

As mentioned in section B.1.5, a practical solution to compute dense disparity maps in real-time can be implemented by using the epipolar geometry of stereo images in combination with a correlation based algorithm. We have implemented such a method, that consists of calculating a similarity measure, say the sum of the absolute difference of the values of the pixel in the neighbourhood of the two points under consideration, based on the intensity of the pixels. The most similar pixels in different images are considered to be matched. The speed requirement is achieved using two schemes. To reduce the amount of calculations, images are first standardised (such that 2 corresponding pixels are in the same row in both images) using stereo calibration. A window shifting technique [131] speeds up the computation by eliminating redundant computations.

Figure B.7 shows a classical pair of standardised images from the Tsukuba university repository and the corresponding disparity map obtained with our implementation of a correlation based algorithm. The red areas represent the pixels that are occluded. The lighter the colour the nearer the entity is to the camera.

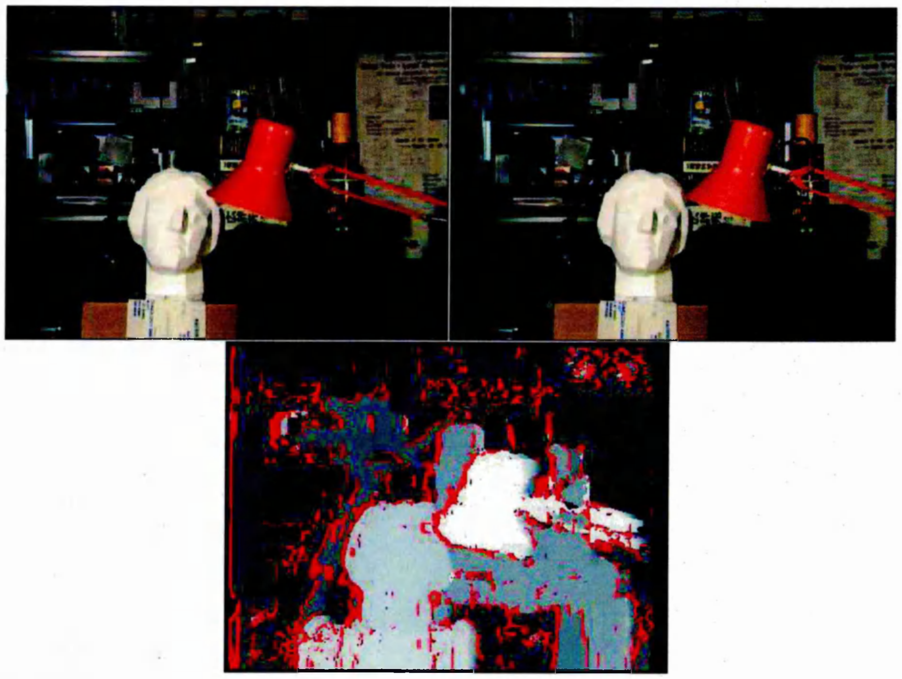


Figure B.7: Result of our implementation on a well known pair of standardised images from the Tsubaka university repository.

Figure B.6 illustrates problems due to illumination. On the disparity map the computer box in the scene contains black regions wrongly suggesting that there are holes in the surface. Further analysis of the original image leads us to suggest that the light reflection dissimilarity due to the slight difference of the position of the camera is responsible for the erroneous result.

Figure B.8 presents different depth maps with a checker board placed at different positions.

B.4 Triangulation

All the necessary information to locate a point relative to the stereo vision system is now known. The calibration of each of the cameras allows the estimation of the coordinate of any line passing through the camera origin and a pixel. Moreover, stereo calibration gives the relative orientation and position of the two cameras. Therefore, knowing the corresponding pixel of an object point allows the complete determination of the triangle whose vertices are the two camera origins and the the object point.

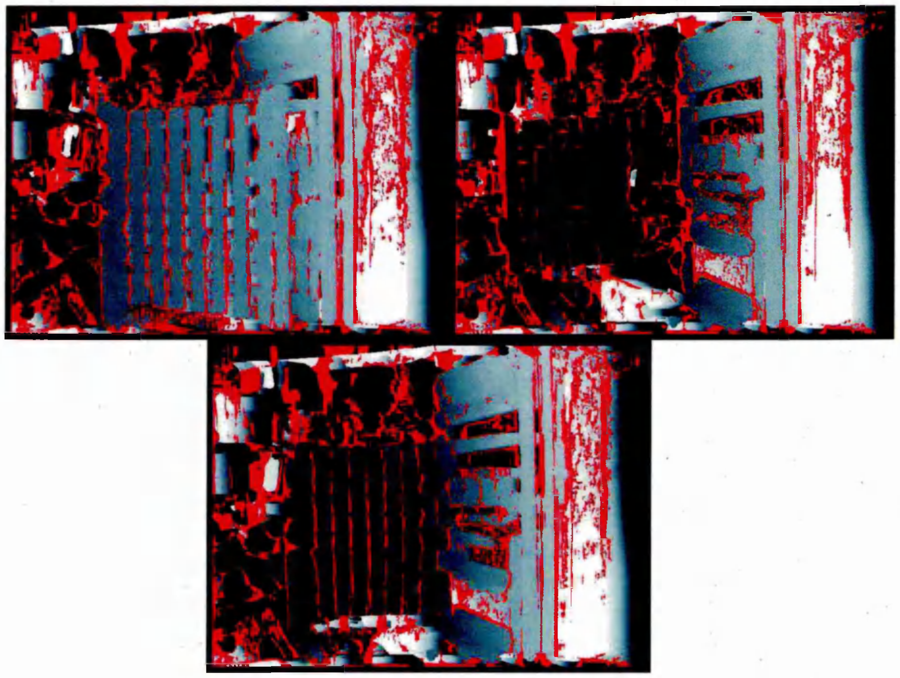


Figure B.8: Different disparity maps using our calibrated images.

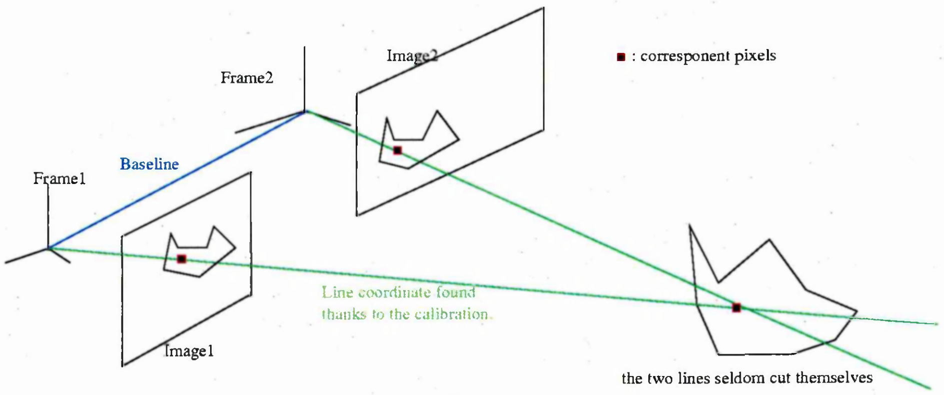


Figure B.9: Triangulation of a point.

In practice, noise and discreteness of the data results in a triangle that presents an open angle as shown in figure B.9. The location of the object can be considered to be the nearest point to both lines.

B.5 Summary

Although stereo-vision has naturally emerged for biological "systems" and proved to be useful in various applications, such as segmentation or in robotics for obstacle avoidance, for systems aiming at capturing the geometry of an object it is impaired with:

1. unreliability when the object has insufficient texture to correctly evaluate the depth map.
2. imprecision since depth accuracy is decreasing proportionally to the inverse of the distance between the camera system and the entity position.
3. slowness due to computationally expensive techniques to evaluate the depth map. Although this is not an issue anymore thanks to technique like graph cuts.

Creating a model of an object is better tackled using structured light, like a laser or a projector, or time of flight [113] methods. Therefore, stereo vision has to be considered with circumspection for model construction of objects since other technology can achieve better results by evaluating the 3-D geometry of a visible scene, in a more accurate, efficient and precise manner.

Appendix C

Further testing of the particle filter

C.1 Table tennis sequence

Figures C.1 and C.2, show the results of the tests of the implementation of the particle filter on 2 ping-pong sequences. In this case a correlation measure and the modified particle filter that takes into account the speed of the object were used. Some parameters, such as the distance of relocation of particles for the propagation step of the particle filter, needed to be tuned but this was done quickly and the tracking worked almost immediately.

C.2 Rubik's cube sequence

In order to make sure the implementation was robust and generic enough a third tracking scenario was selected: the 3-D tracking of a Rubik's cube. The measure associated with this scenario differed from previous tracking scenarios since a 3-D wireframe model was used instead of a template image of the object.

Brown *et al.* [132] proposed a method to acquire a 3-D wireframe model using a semi-automatic method. Outlines of the tracked object are drawn on the image. It is then tracked using a Kalman filter, the accumulated information then serves to determine the wireframe model. The method also explains how lines can be tracked in 3-D. For more references on 3-D tracking and 3-D models see page 179.

In a first attempt to track the Rubik's cube the outline of the cube was projected on the image and the number of edge points close to this outline were counted. Figure

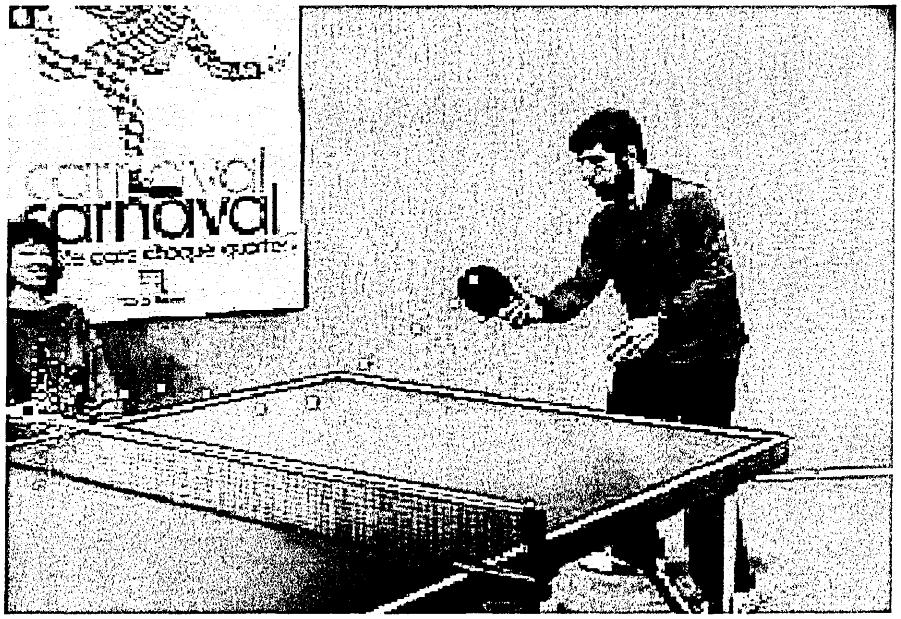


Figure C.1: Tracking of a ping-pong ball. Top left corner, the ball template image. The large white dots represent the previous tracked positions of the ball. The large black dot, the current position of the ball and the small white dots, the particles' positions.

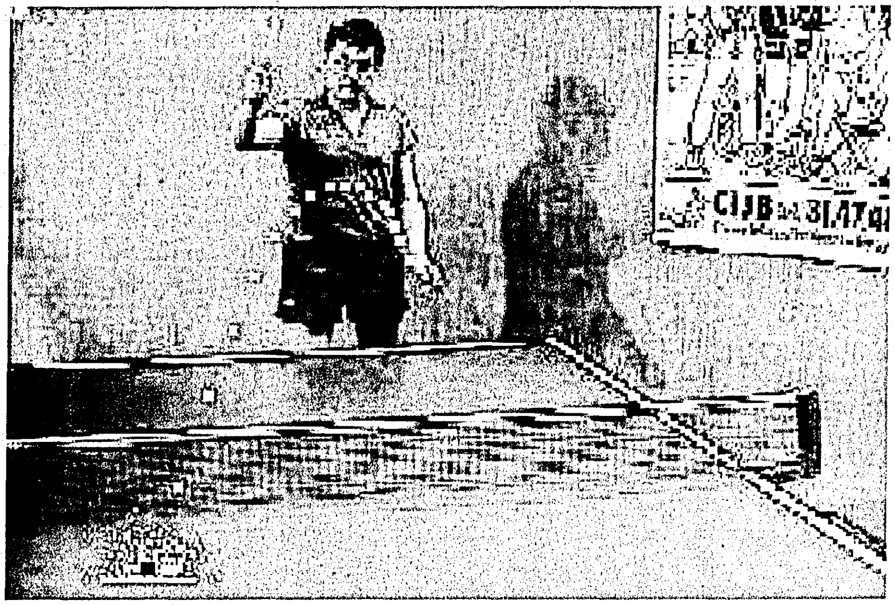


Figure C.2: The large black square is the probable actual position of the ping-pong ball. The small white dots indicate the particles' positions and the larger white squares represent the earlier tracked positions.

C.3 shows that the tracking failed quickly.

To improve our model a hand-crafted 3-D wireframe model of the Rubik's cube

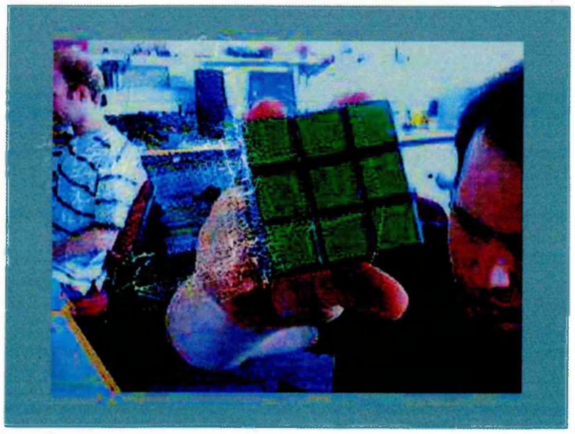


Figure C.3: After a few frame the tracking fails completely.

was made by determining the 3-D coordinate of the 54 square elements lying at the surface of the Rubik's cube. Edges corresponding to the boundary of these squares were used to represent the Rubik's cube. A projective model of the camera was used to project these lines on the image and comparison with an edge filter image of the sequence were used to validate hypotheses. Although the particle cloud appeared to stay focused longer around the Rubik's cube, the particles still spread quite quickly resulting in the tracking failing again.

In order to improve the filtering method the following was undertaken. First, a coarse colour filter was applied to the image. Morphological operators, opening and closing, were used to roughly identify the position of the Rubik's cube. In indsight the use a colour filter based on the hue saturation ligthness (*HSL*) colour space would be preferable and morphological operator parameters could have been better tuned to eliminate the unnecessary square appearing in the third image of figure C.4 and which does not correspond to the Rubik's position. This first segmentation was then used as a stencil to select the edges filtered with a Canny edge detector. Then a distance map was created such that each pixel indicates the distance to its closest edge. This allows the application of minimisation techniques to better position the model on the image as described further down. Figure C.4 illustrates these steps. Figure C.5 shows this distance map in false colour.

The next improvement was to remove hidden lines from the measurement data. This is a well-known problem discussed at length by the computer graphics community. The literature review shows that there is no easy algorithm that allows the determination of which edges are occluded by the surface of an object. Hidden line

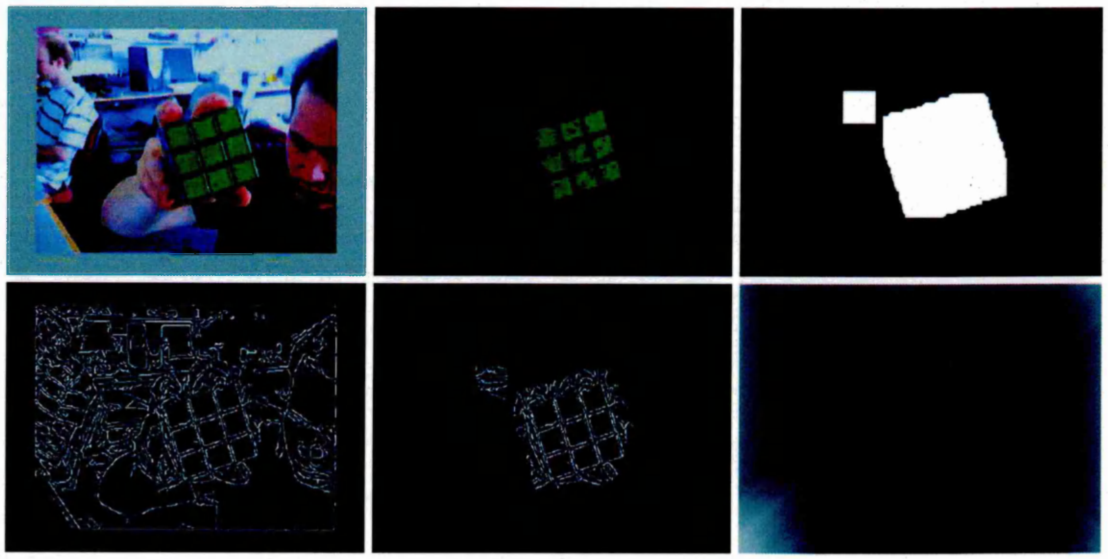


Figure C.4: Different stage of the image filtering process

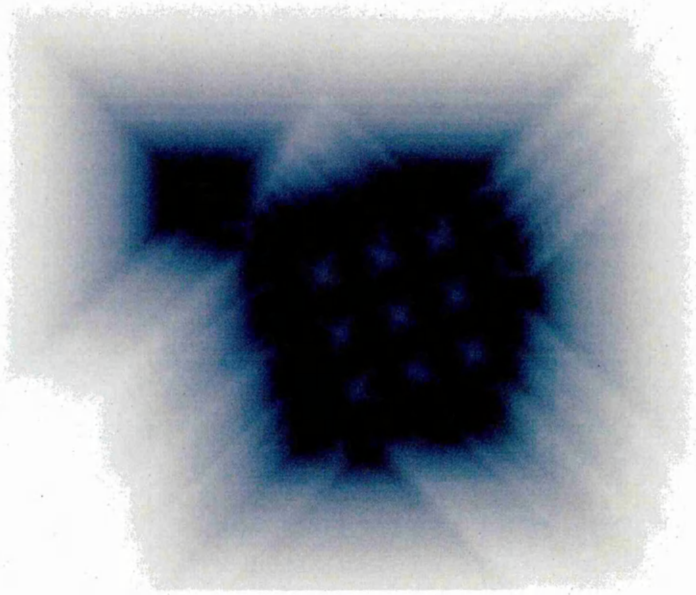


Figure C.5: The filtered image with false colours.

removal is considered as one of the most "tedious tasks in 3-D programming" [133]. An analysis of the paper [134] shows that there is a huge number of special cases to take into account that largely justifies this statement. Some implementations can be found, nevertheless they do not exhaustively remove hidden lines, culling of back-face polygon is generally the only operation done. While the algebraic approach proved to be difficult, hardware implementations using a Z-buffer technique (also called stencil [135] buffer) to implement efficiently hidden line removal are readily available. Lines are written in the buffer and overwritten with the background colour when another polygon hides them. OpenGL[1] has been used to render the corresponding cube to a particle. A major issue of this technique is that primitives are lost during the rendering process. Thus to obtain the outline of the cube the image was rendered and then filtered to obtain the points corresponding to edge locations. This process is slow and directly obtaining the visible edges primitives would have been more desirable. For further information, [136] presents a taxonomy of different hidden line removal algorithms.

Despite this last improvement, the tracking continued to fail as shown in figure C.6. The presented frames indicates that one of the reasons was the clustering of particles around positions that match some of the Rubik's edges but not the correct ones. The technique discussed in section 3.5.4 that involves using a complementary measure could be implemented using the colour segmentation based measure. However a different approach was utilised. In order to reduce the number of particles used and thus increase the tracking speed, a gradient descent implementation technique was employed. The particles having promising results were relocated using a gradient descent algorithm in order to maximise their weight. Although this improved the tracking, it did not completely succeed. However, the successive implementations allowed us to refine the genericalness of the particle filter implementation.

After further consideration of the problem, the following method was conceived: using the Hough transform some of the main line of the cube could be determined. The intersection of the lines could then provide some of the points of the Rubik's cube at the intersection of the Rubik's cube edges. Point correspondence, using a

[1]A standard used by graphic cards to render 3-D surfaces.

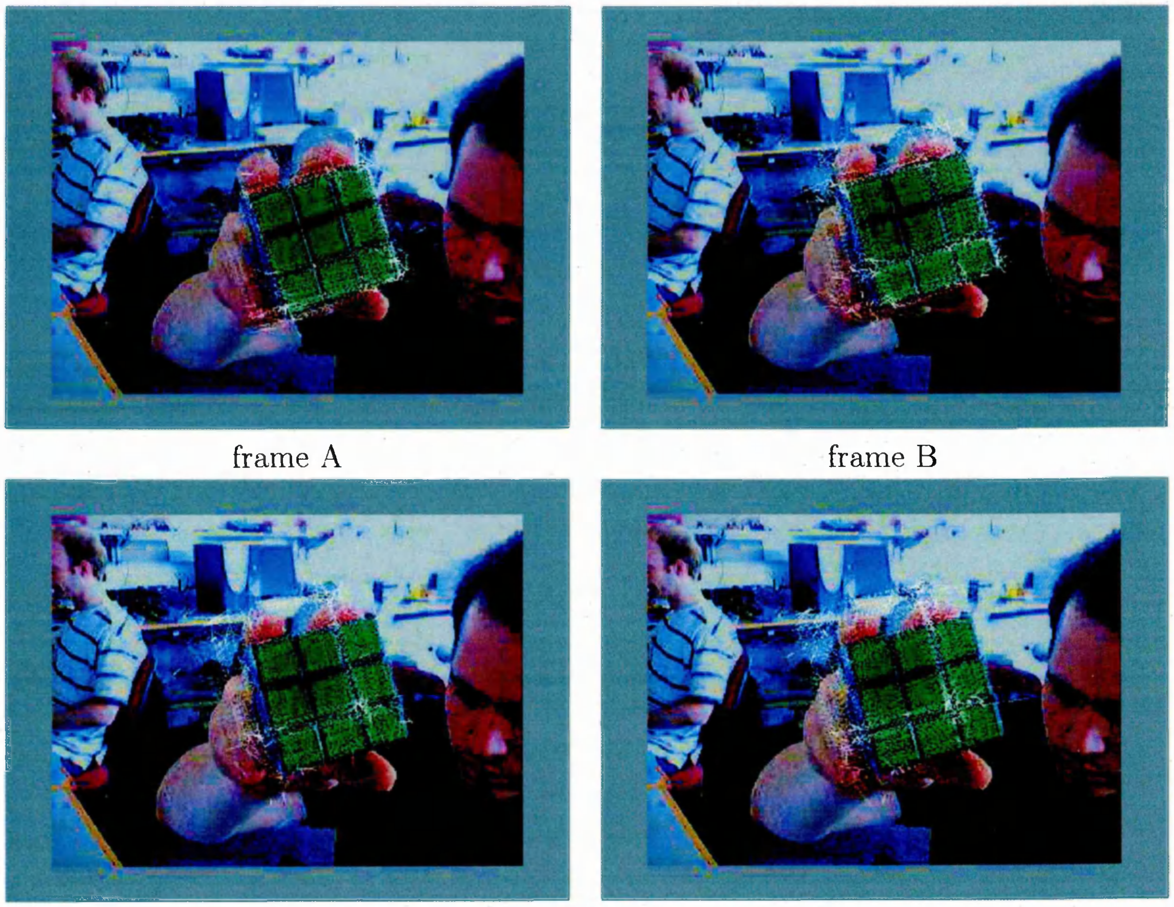


Figure C.6: Tracking a Rubik's cube, transparent white doted lines indicates the hypothesis location.

RANSAC[2] method would ultimately provide the position of the Rubik's cube. Although this method does not involve particle filters it enabled a better acquaintance with the Hough transform method. This was eventually used during the MiCRoN project and constituted the starting point for the research in template reduction presented in previous chapters.

[2]Random sampling consensus, see page 179

Appendix D

Machine vision and computer vision

The distinction between machine vision and computer vision is often misunderstood and a short explanation is given. The various techniques developed in "machine vision" usually tend to be related to the needs of industry and includes the design of the physical vision system (camera, light source, laser, lenses, interface, processing unit *etc.*) in an industrial environment as well as evaluating risks associated to the system. In contrast "computer vision" is more concerned with the algorithmic issues and is considered to be a sub-field of artificial intelligence [137]. The two fields are closely interrelated and the boundary is blurred. However it is often the case that machine vision development is industry based and computer vision academically based.

Computer vision comprises a vast body of knowledge that intersects with diverse scientific fields (computer science, psycho physics, mathematics, engineering) and also has its own sub-development, *e.g.* the theory of camera calibration. The number of techniques and ideas it encompasses is so vast that it is difficult to intuitively grasp this field in its totality. Some idea of its extent can be gained by browsing *CVonline*, an evolving, distributed, non-proprietary, on-line compendium of computer vision. Many researchers have claimed that computer vision has come of age. Recent technical improvements (support vector machines, the generalised usage of statistical methods *etc*) along with developments related to the Internet have led experts to suggest that, despite the incredible complexity of the task, it might be possible to mimic the vision capability of the brain for scene interpretation.

It has been estimated that roughly 60 percent of the human brain cortex is involved in vision processes.

Appendix E

Tracking source code

A minimalistic implementation of the tracking algorithm follows. It is also available in the Mimas library in the examples/tracking/stencil_minimalist directory. The header file:

```
#ifndef MMVL_STENCIL_TRACKING_HH_INCLUDED
#define MMVL_STENCIL_TRACKING_HH_INCLUDED

#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <set>
#include <algorithm>
#include <boost/multi_array.hpp>
#include <boost/numeric/ublas/vector.hpp>

#include <image_fileinput.h>
#include <image_fileoutput.h>
#include <image.h>


/**
 * Feature-image has to be given in a coordinate sytem centered
 * on the middle of the previous position of the object.
 *
 * Inspired from the bounded hough transform.
 *
 * "Efficient Tracking with the Bounded hough Transform.
 * Michael Greenspan, Limin Shang, Piotr Jasiobedzki"
 *
 *As it might be long to pre-calculate the data structure
 *necessary for the tracking it would be interesting
 *to save the object once it has been instantiated and initialised.
 *At the moment the best solution seems to use
 *serialization. This is provided by
 *boost serialization available in version 1.32 of the boost library
 *
 *
 *How to take into account other clues like color, feature orientation...
 *
 * Time-stamp: <2005-01-28 12:10:45 engmb>
 * Copyright (C) 2005 by Manuel Boissenin
 *
 * @author Manuel Boissenin
 * @date 17/01/2005
 *
 **************************************************************************/

//interval_type can be continuous or discrete (for stack of images)

class Interval{
public:
  enum interval_type {Continuous = 0, Discrete};
  enum interval_type type;

  //for Continous type
  float lower_bound;
  float upper_bound;

  //for Discrete type
  int value;

  Interval(float lowerBound, float uperBound)
    :type(Continuous), lower_bound(lowerBound), upper_bound(uperBound)
  {};

  Interval(int value)
    :type(Discrete),value(value)
  {};
};


/**
* A transformation consists of a closed subset of the transformation
* space.
*
* For the stencil tracking the idea is partition the transformation space.
* The resulting discrete sets have to be small enough to consider that
* any of their elements is within the tolerence error of the average of
* the subset elements.
***************************************************************************/

class Transformation{
public:
  std::vector<Interval> intervals;
  int votes;

  Transformation():
    votes(0){}

struct lttransf : public std::binary_function<Transformation, Transformation, bool> {
    bool operator()(Transformation x, Transformation y) { return  x.votes < y.votes;}
  };

/**
*Feature is used to describe the x,y position of a feature
*
*ltfeature is used to compare two features by std::set
*
* @author Manuel Boissenin
* @date 18/01/2005
***************************************************************************/

typedef boost::numeric::ublas::vector<int> Feature;

struct ltfeature
{
  bool operator()(const Feature &f1, const Feature &f2) const
  {
    if (f1(0) == f2(0))
      {
        return f1(1) < f2(1);
      }
    return f1(0) < f2(0);
  }
};

class stencil_tracking {
public:

  //Put the frame of the model in the middle of the image.
  //Features are given in a model centered coordinate system.
  //We assume that all model images are of the same size and
  //that the object has the same orientation
  //and is centered in the middle of the image.

  stencil_tracking( const std::vector<Feature>  &feature_image,
                    int model_width, int model_height,
                    int x_min, int x_max, int x_sub,
                    int y_min, int y_max, int y_sub);

  //This is used as an indicator (when compared with the number of
  //hit find) for the success of the tracking
  int number_of_features;

/*
  Pseudo-code

  fill_image_space()
  {
  for each transformation
  for each feature of the image corresponding to the transformation
    {
      calculate locus of point according to the sub-transformed space;
      add transformation reference to this locus to the image_space array;
    }
  }
*/

//this is separated from the constructor as the data
//structure it fills might be latter loaded
  void fill_image_space(void);

  //The feature should be expressed in a object centered coordinate system of
  //the previous tracked location.
  Transformation track(std::vector<Feature> &features);

  int get_model_width(void) const { return model_width; }
  int get_model_height(void) const { return model_height; }

private:
  int model_width;
  int model_height;
  int x_sub;
  int y_sub;


  //To store the vector of feature of an image
  std::vector<Feature>  feature_image;

  boost::multi_array<Transformation,2> transformation_space;
  //to index the transform space
  typedef boost::multi_array<Transformation,2>::index idx_ts;

  //size of the image space depends on the image model size and
  //the transformation space.
  boost::multi_array<std::vector<Transformation *>,2> image_space;


  std::vector<Transformation *>::iterator transf_it;
  //to index the image space
  typedef boost::multi_array<std::vector<Transformation *>,2>::index idx_is;
```

```
};


typedef boost::shared_ptr< stencil_tracking > stencil_tracking_ptr;

#endif
```

The stencil_tracking.cc file:

```
/**
 * A minimalist code to help understand the essential
 * elements of stencil tracking.
 *
 *
 * Time-stamp: <05/10/19 21:06:11 engmb>
 * Copyright (C) 2005 by Manuel Boissenin
 *
 * @author Manuel Boissenin
 * @date 17/01/2005
 *
 *********************************************************************/
#ifndef NDEBUG
#include <boost/multi_array.hpp>
#endif
#include <utility>
#include "stencil_tracking.h"

#ifndef NDEBUG
using namespace boost;
#endif

stencil_tracking::stencil_tracking( const std::vector<Feature> &feature_image,
                                    int model_width, int model_height,
                                    int x_min, int x_max, int x_sub,
                                    int y_min, int y_max, int y_sub):
  model_width(model_width),
  model_height(model_height),
  x_sub(x_sub), y_sub(y_sub),
  feature_image(feature_image),
  transformation_space(boost::extents[x_sub][y_sub]),
  image_space(boost::extents[model_width + (x_max - x_min) ][model_height + (y_max - y_min)])
{
  float x_size = float(x_max - x_min)/ x_sub;
  float y_size = float(y_max - y_min)/ y_sub;

  //Fill the transformation space with its values
  for(idx_ts x = 0; x < x_sub; x++)
    for(idx_ts y = 0; y < y_sub; y++)
          {
            std::vector<Interval> tmp_interv;
            tmp_interv.push_back(Interval(x_min + x * x_size, x_min + (1+x)*x_size));
            tmp_interv.push_back(Interval(y_min + y * y_size, y_min + (1+y)*y_size));

            Transformation tmp_transf;
            tmp_transf.intervals = tmp_interv;

            transformation_space[x][y] = tmp_transf;
          }

    number_of_features = feature_image.size()      ;

}


/*
  Pseudo-code

  fill_image_space()
  {
```

```
  for each transformation
     for each feature of the image corresponding to the transformation{
         calculate locus of point according to the sub-transformed space;
         add transformation reference to this locus to the image_space array;
     }
  }

this function isn't called by the constructor in the hope later to serialize the
tracker (it can be very long to calculate the data structure especially for high
dimensional spaces)
*/

void stencil_tracking::fill_image_space(void){
#ifndef NDEBUG
  unsigned n = 0;
#endif

    for(idx_ts x = 0; x < x_sub; x++)
      for(idx_ts y = 0; y < y_sub; y++){

        Transformation *tmp_transf = &transformation_space[x][y];

        std::set<Feature, ltfeature> x_trans_locus;
          for(int x = int(tmp_transf->intervals[0].lower_bound);
              x <= tmp_transf->intervals[0].upper_bound; x++)
            {
              for( std::vector<Feature>::const_iterator feat_it = feature_image.begin();
                   feat_it != feature_image.end(); feat_it++)
                {
                  Feature tmp_feature(2);
                  tmp_feature(0) = (*feat_it)(0) + x;
                  tmp_feature(1) = (*feat_it)(1);
                  x_trans_locus.insert(tmp_feature);
                }
            }

          std::set<Feature, ltfeature> xy_trans_locus;
          for(int y = int(tmp_transf->intervals[1].lower_bound);
              y <= tmp_transf->intervals[1].upper_bound; y++)
            {
              for( std::set<Feature, ltfeature>::const_iterator feat_it =
                  x_trans_locus.begin();
              feat_it != x_trans_locus.end(); feat_it++)
                {
                  Feature tmp_feature(2);
                  tmp_feature(0) = (*feat_it)(0);
                  tmp_feature(1) = (*feat_it)(1) + y;
                  xy_trans_locus.insert(tmp_feature);
                }
            }

          //fill the image space with the pointer to the transformation
          for( std::set<Feature, ltfeature>::const_iterator feat_it = xy_trans_locus.begin();
               feat_it != xy_trans_locus.end();
               feat_it++)
            {
              //transform back to top left coordinate system
              int x = (*feat_it)(0) + image_space.shape()[0]/2;
              int y = image_space.shape()[1]/2 - (*feat_it)(1);

              //Not to add twice the same transform if
              //the same sub-transformation bring two points
              //to the same place

              if((image_space[x][y].empty()) || (image_space[x][y].back() != tmp_transf))
                {
                  image_space[x][y].push_back(tmp_transf);
                }
            }
        }

}

Transformation stencil_tracking::track(std::vector<Feature> &features)
{
  //reinitialize the number of vote
  for(unsigned int n = 0; n < transformation_space.num_elements(); n++)
    transformation_space.data()[n].votes = 0;

  for( std::vector<Feature >::const_iterator feature_it = features.begin();
       feature_it != features.end(); feature_it++)
    {
      Feature temp(2);
      temp(0) = (*feature_it)(0) + image_space.shape()[0]/2;
      temp(1) = image_space.shape()[1]/2 - (*feature_it)(1);

      for(transf_it = image_space[temp(0)][temp(1)].begin();
          transf_it != image_space[temp(0)][temp(1)].end();
          transf_it++)
        (*transf_it)->votes++;
    }

  Transformation max_element = *(std::max_element(transformation_space.data(),
                                                  transformation_space.data()+
                                                  transformation_space.num_elements(),
                                                  lttransf()));

  return max_element;
}
```

Part of the main file. The whole implementation is provided with the Mimas examples in the tracking/stencil_minimalist repertory.

```
int main(int argc, char** argv)
{
  int retVal = 0;
  try {
    glutInit( &argc, argv );
    x11_display disp;
    image_mesaoutput<unsigned char > display( &disp );

    //begin load parametres

    string model_image = argv[11];
    string base_name_image = argv[12];

    image<unsigned char> current_image;

    mimas::hf::image_loader<unsigned char> im_ld(current_image ,
    mimas::hf::image_loader<unsigned char >::set(base_name_image,4,".pgm",0,299));
    int threshold = atoi(argv[13]);

    int model_width = atoi(argv[1]);
    int model_height = atoi(argv[2]);
    int center_x = atoi(argv[3]);
    int center_y = atoi(argv[4]);


    object_position current_position;
    current_position.x = center_x;
    current_position.y = center_y;

    int x_min = atoi(argv[5]);
    int x_max = atoi(argv[6]);
    int x_sub = atoi(argv[7]);

    int y_min = atoi(argv[8]);
    int y_max = atoi(argv[9]);
    int y_sub = atoi(argv[10]);


    std::vector< Feature > model = create_model(model_width, model_height,
                                                center_x, center_y, model_image, threshold);

#ifndef NDEBUG
    display_features(model, &disp);
#endif
    //the class can treat more complex model (stack of images).

    stencil_tracking tracker( model, model_width, model_height,
                              x_min,x_max,x_sub,
                              y_min,y_max,y_sub);

    tracker.fill_image_space();

    while(true)
      {
        im_ld.next();
        std::vector< Feature > centered_features =
        get_features(model_width + x_max - x_min ,
                     model_height + y_max - y_min,
                     current_position.x, current_position.y,
                     current_image, threshold);

#ifndef NDEBUG
        display_features(centered_features, &disp);
#endif

        Transformation result = tracker.track(centered_features);
        float percentage = float(result.votes) / tracker.number_of_features;
#ifndef NDEBUG
        cerr<<"Indicator of success is "<<percentage<<endl;
#endif

        current_position.x +=
                              (int)((result.intervals[0].lower_bound +
                              result.intervals[0].upper_bound) / 2.0 );
        current_position.y +=
                               (int)( (result.intervals[1].lower_bound +
                               result.intervals[1].upper_bound) / 2.0 );

        current_image.setPixel(current_position.x, current_position.y, 255);
        display << current_image;

      }
  } catch ( exception &e )
    {
      cerr << "An exception occurred: " << e.what() << endl;
      retVal = 1;
    }
  return retVal;
}
```

Appendix F

Extended abstract on the stencilled Hough transform

Shape information: using state space to select discriminative configuration of points

Manuel Boissenin, Bala P. Amavasai, Jan Wedekind, Reza Saatchi
Microsystems and Machine Vision Laboratory,
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, United Kingdom
`http://www.shu.ac.uk/mmvl`
`Manuel.Boissenin@gmail.com`

Abstract – *A novel approach for the analysis of shape is proposed. A shape is hither considered to be a set of points. The idea consists in analysing a shape relatively to the set of transformations the shape can undergo. The set of transformation is also referred as state space. From this analysis it is concluded that some points contribute more information about the shape than others. A framework to define and quantify the information contributed by set of points is developed in here. By doing so, it is then possible to decimate a set of points to obtain a reduced template of a shape. The proposed method has positive consequences for tracking and recognition algorithms.*

I Locating a shape on an image

The information implicitly contributed by a shape might be defined according to the point configurations that are needed to determine its characteristics (*e.g.* position and scale).

By identifying a subset of points of a shape that characterises a shape, an algorithm to locate and track an object can be optimised to reduce the memory usage and improve their speed performances. We focus on quantifying the information contributed from the relative position of unidentifiable points, extracted with an edge detector for instance, and the state space under consideration.

While studying this problem the following questions arise: for a given set of points what criteria can be used to determine a subset of points that uniquely characterises the shape? Given a set of points, is it possible to quantify the likelihood of the position of a shape that matches image points? In other words, given a set of image points, is it possible to evaluate the probability density function (*pdf*) of the shape state? Can points extracted from an image be determined to be part of a random set of points or of a given shape? Is it possible to quantify the information contributed by individual points to a set of points?

In order to answer these questions the following notation are used:

- the shape S which is a set of points,
- the transformation space T, or state space, is another set representing the possible transformation that the shape can undergo.
- $p_i \in I, i = 1 \ldots n$, is a set of n points belonging to the image $I \subset \mathbb{R}^2$

Considering a set of points in the image space, we quantify the transformations that are compatible with putting into correspondance points from the shape with points of the image. The smaller this set of transformations is, the more information the set of point is contributing to the knowledge of the shape state. Formally, we consider the set $\{t | \forall p_i \in I, \exists s_j \in s \subset S, t(s_j) = p_i, t \in T\}$

This shift in perspective that consists in considering the transformation space is the classical one that is used by Hough transform [2] [4] [3] related algorithms. A point on the image space is selected and according to the transformation space and the shape under consideration all possible transformations that are compatible with bringing a point of the shape in correspondence with the image point get a vote. After considering a certain number of points in the image space, the transformations getting almost the same number of votes are transformations that bring the points of S in correspondence with almost all the point of the image that were considered. With additional considerations we propose the following definition:

Definition 1 *Let T be the transformation space, $\mathrm{B}_\delta(p)$ a sphere of the same dimension than T, of radius δ and of centre p. P a set of points of the image, S the set of points of the shape, $C(P)$ the set of transformations compatible with the points of the image. We define the quantity of transformations $Q(P)$ contained in $C(P)$ as:*

$$C(P) := \{t \in T | \mathrm{card}(t(S) \cap P) = \mathrm{card}(P)\}$$

$$\mathcal{Q}_\delta^T(P) := \min \mathrm{card}(\{\mathrm{B}_\delta(p) | p \in T, \delta \in \mathbb{R}, \mathrm{B}_\delta(p) \cap C(P) \neq \emptyset\}) \tag{1}$$

A set of points P is said to uniquely characterises a shape transformation relative to a transformation space T and with an error δ if $\mathcal{Q}_\delta^T(P)$ is equal to 1.

We now consider a set of points S extracted from a shape and we consider that T is bounded. The following quantifies the information contributed by a set of points of a shape relative to a transformation space T and an accuracy of δ.

$$\mathcal{I}_\delta^T(S) := \sup_{t \in T} \mathcal{Q}_\delta^T(t(S)) \tag{2}$$

The number of configurations to consider becomes huge very quickly: for k point configurations out of n points of the shape the number of cases to consider is $\binom{n}{k}$, even though, possibly, only a small set of points are sufficient to represent the object and therefore k should be small, this may be quite large. Monte Carlo methods and genetic algorithms are good candidates to find configurations that are close to optimal. Implementation details to evaluate $\mathcal{I}(S)$ are given in the following section.

II Evaluation of the quantity of information contributed by a set of points

A methodology to evaluate the quantity of information contributed by a set of points P from an image is now discussed.

A bounded transformation space T is divided into small hypercubes $h \in H$ such that H is a partition of T. We note $t(S) := \{t(s) | s \in S\}$. A is a 2-D array of the size of the image. $a_{i,j}$ are the elements of A and correspond to the points overlapped by the pixel on line i and column j noted $I_{i,j}$.

For each set of points $h \in H$ and for each $t \in h, t(S)$ is evaluated and a reference to h is stored in $a_{i,j}$ whenever $I_{i,j} \cap t(S) \neq \emptyset$. This operation is not completely trivial to implement and needs some approximation to be done in a reasonable amount of time, however time is not critical since these operations can be done offline. Once this is done, $a_{i,j}$ contains the references to the hypercubes that contain the transformations that shift a point of the shape to the corresponding pixel $I_{i,j}$. We note $h^k_{i,j}, k = 1..n_{ref}$ the hypercubes referenced in $a_{i,j}$.

It is now shown that the number of references n_{ref} that is contained by $a_{i,j}$ is equal to $f\mathcal{Q}^T_\delta(I_{i,j})$ with $a \leq f \leq b$ where $(a, b, f) \in \mathbb{R}^3$, a and b are two constants that depends on the size of the hypercubes and the error sphere.

All hypercubes are assumed to have the same size but the demonstration holds with hyper-volumes of different size. One just has to consider the extremal cases. If the minimal number of spheres needed to cover one hypercube is m then at worst the minimal number of spheres to cover all hypercubes would be $m \cdot n_{ref}$ thus $\mathcal{Q}^T_\delta(I_{i,j}) \leq m \cdot n_{ref}$. If a sphere can intersect at most p hypercubes then we need at least $\frac{n_{ref}}{p}$ spheres to cover all transformations that are in the hypercubes and thus $\frac{n_{ref}}{p} \leq \mathcal{Q}^T_\delta(I_{i,j})$.

Lemma 1 *$\exists(a, b) \in \mathbb{R}^{*2}_+$, such that $a \cdot \mathcal{Q}^T_\delta(I_{i,j}) \leq n_{ref} \leq b \cdot \mathcal{Q}^T_\delta(I_{i,j})$*

The most unfavourable case being when a set of transformations, that can be covered by a unique sphere, lie on the interface of multiple hypercubes. If the dimension of the transformation space is n, the number of hypercubes that cover the set of transformation can be as high as 2^n. As a consequence, in practise, not only must n_{ref} be considered but also if the hypercubes are contiguous or not.

Therefore, it is possible to have an evaluation of the quantity of the transformations compatible with a set of points P by considering:

$$\mathcal{Q}^T_\delta(P) \sim \text{card}\Big(\bigcap_{\substack{k,\\ i,j | I_{i,j} \in P}} h^k_{i,j} \Big) \tag{3}$$

and as a consequence the quantity of information of a subset of points S' of a shape can be estimated by:

$$\mathcal{I}^T_\delta(S') \sim \sup_{t \in T} \text{card}\Big(\bigcap_{\substack{k,\\ i,j | I_{i,j} \cap t(S') \neq \emptyset}} h^k_{i,j} \Big) \tag{4}$$

which is much more computationally expensive to evaluate. In order to reduce computation it is suggested that sampling the transformation space might give a good approximation: by selecting a few transformations from each hypercube for instance. Proofs, experiments and more theoretical studies remain open to research.

III Conclusion

We proposed a measure to quantify the information contributed by a subset of point of a shape. By shape we refer to any set of points associated to a transformation space. We then proposed an algorithm to evaluate this measure. The algorithm allows to identify subsets of points of a shape that characterise uniquely its state for a given transformation space (perspective transformations for instance). The technique has been successfully combined with a tracking algorithm that uses decimated template with significant improvement for its speed (by a factor of 6000 in a specific instance) and memory usage. For reason of space, results can not be developed here.

It has been argued that high curvature points [5][1] conveys more information on a shape than other points. The presented framework, that takes into account the state space of a shape, provides a way to quantify this.

References

[1] F. Attneave and M. Arnoult. The quantitative study of shape and pattern perception. *Psychological Bulletin*, 53(6):452–471, 1956.

[2] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 714–725. Kaufmann, Los Altos, CA., 1987.

[3] M. Greenspan, L. Shang, and P. Jasiobedzki. Efficient tracking with the bounded hough transform. In *CVPR '04: Computer Vision and Pattern Recognition*, pages 520–527, 2004.

[4] V. F. Leavers. Which hough transform? *CVGIP: Image Underst.*, 58(2):250–264, 1993.

[5] S. Loncaric. A Survey of Shape Analysis Techniques, Pattern Recognition. *Bd*, 31:983–1001.

References

[1] Champaneria, A. Parallelizing the condensation algorithm for visual tracking. [online, last accessed October 2007], 2002. http://beowulf.lcs.mit.edu/18.337-2002/projects-2002/amay/partracker/doc/.

[2] Welch, G. and Foxlin, E. Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, pages 24–38, 2002.

[3] The MINIMAN project. [online, last accessed October 2007]. http://vision.eng.shu.ac.uk/mmvlwiki/index.php/Miniman.

[4] The MiCRoN project. [online, last accessed October 2007]. http://i60p4.ira.uka.de/tiki/tiki-index.php?page=MiCRoN.

[5] The Mimas library. [online, last accessed October 2007]. http://vision.eng.shu.ac.uk/mediawiki/index.php/Mimas.

[6] Greenspan, M., Shang, L., and Jasiobedzki, P. Efficient tracking with the bounded Hough transform. In *CVPR '04: Computer Vision and Pattern Recognition*, pages 520–527, 2004.

[7] Blake, A. and Isard, M. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion.* Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.

[8] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[9] Malladi, R., Sethian, J. A., and Vemuri, B. C. Shape modeling with front propagation: a level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[10] Brown, Lisa, G. A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325–376, 1992.

[11] Viola, P. and Jones, M. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[12] Jain, A. K., Mao, J., and Mohiuddin, K. M. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996.

[13] Ramamoorthi, R. and Arvo, J. Creating generative models from range images. *Proceedings of SIGGRAPH*, 99:195–204, 1999.

[14] Zitova, B. and Flusser, J. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

[15] Loncaric, S. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[16] Walker, K. N., Cootes, T. F., and Taylor, C. J. Automatically building appearance models from image sequences using salient features. *Image and Vision Computing*, 20(5):435–440, 2002.

[17] Cootes, T. F., Edwards, G. J., and Taylor, C. J. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[18] Cootes, T. F. and Taylor, C. J. Statistical models of appearance for computer vision. *World Wide Web Publication, February*, 2001.

[19] Piccardi, M. Background subtraction techniques: a review. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.

[20] Sugrue, M. and Davies, E. R. Motion distillation for pedestrian surveillance. *The Sixth IEEE International Workshop on Visual Surveillance*, 2006.

[21] Ronse, C. A lattice-theoretical morphological view on template extraction in images. *Journal of Visual Communication and Image Representation*, 7(3):273–295, 1996.

[22] Naegel, B., Passat, N., and Ronse, C. Grey-level hit-or-miss transforms–Part I: Unified theory. *Pattern Recognition*, 40(2):635–647, 2007/2.

[23] Naegel, B., Passat, N., and Ronse, C. Grey-level hit-or-miss transforms–Part II: Application to angiographic image processing. *Pattern Recognition*, 40(2):648–658, 2007/2.

[24] Soille, P. On morphological operators based on rank filters. *Pattern Recognition*, 35(2):527–535, 2002.

[25] Gasteratos, A. and Andreadis, I. Soft mathematical morphology: extensions, algorithms and implementations. *Advances in Imaging and Electron Physics*, 110(1):63–99, 1999.

[26] Nachtegael, M. and Kerre, E. E. Connections between binary, gray-scale and fuzzy mathematical morphologies. *Fuzzy Sets and Systems*, 124(1):73–85, 2001.

[27] Baumann, D. and Tinembart, J. Mathematical morphology image analysis on FPGA.

[28] Liu, L. Morphological hit-or-miss transform for binary and gray-tone image processing and its optical implementation. *Optical Engineering*, 33:3447, 1994.

[29] Gope, C. and Kehtarnavaz, N. Affine invariant comparison of point-sets using convex hulls and Hausdorff distances. *Pattern Recognition*, 40(1):309–320, 2007.

[30] Rucklidge, W. J. Efficiently locating objects using the Hausdorff distance. *International Journal of Computer Vision*, 24(3):251–270, 1997.

[31] Bayro-Corrochano, E. and Ortegon-Aguilar, J. Lie algebra template tracking. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2, 2004.

[32] Fitzgibbon, A. W. Robust registration of 2-D and 3-D point sets. *British Machine Vision Conference*, 2:411–420, 2001.

[33] Drummond, T. and Cipolla, R. Application of Lie algebras to visual servoing. *International Journal of Computer Vision*, 37(1):21–41, 2000.

[34] Borgefors, G. Hierarchical Chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[35] Breuel, T. M. Implementation techniques for geometric branch-and-bound matching methods. *Comput. Vis. Image Underst.*, 90(3):258–294, 2003.

[36] Breuel, T. M. Finding lines under bounded error. *Pattern Recognition*, 29(1):167–178, 1996.

[37] Wedekind, J., Boissenin, M., Amavavasai, B., and Caparrelli, F. Object recognition and real-time tracking in microscope imaging. In *Proceedings of the 2006 Irish Machine Vision and Image Processing Conference (IMVIP 2006)*, 2006.

[38] Mukundan, R. and Ramakrishnan, K. *Moment Functions in Image Analysis: Theory and Applications*. World Scientific, 1998.

[39] Mukundan, R. and Ramakrishnan, K. Fast computation of Legendre and Zernike moments. *Pattern Recognition*, 28(9):1433–1442, 1995.

[40] Hwang, S., Billinghurst, M., and Kim, W. Local descriptor by Zernike moments for real-time keypoint matching.

[41] de Rezende, P. and Lee, D. Point set pattern matching in d-dimensions. *Algorithmica*, 13(4):387–404, 1995.

[42] Camera calibration toolbox for matlab. [online, last accessed October 2007]. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[43] Welch, G. and Bishop, G. An introduction to the Kalman filter. *ACM SIGGRAPH 2001 Course Notes*, 2001.

[44] Julier, S. and Uhlmann, J. A new extension of the Kalman filter to nonlinear systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 3, 1997.

[45] Welch, G. and Bishop, G. The Kalman filter. [online, last accessed October 2008]. http://www.cs.unc.edu/~welch/kalman/.

[46] Stenger, B., Mendonca, P., and Cipolla, R. Model-based hand tracking using an unscented Kalman filter. *Proc. British Machine Vision Conference*, 1:63–72, 2001.

[47] Yoon, Y. *A New Kalman-Filter Based Framework for Fast and Accurate Visual Tracking of Rigid Objects.* PhD thesis, Purdue university, 2006.

[48] Nummiaro, K., Koller-Meier, E., and Van Gool, L. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[49] Okuma, K., Taleghani, A., de Freitas, N., Little, J. J., and Lowe, D. G. A boosted particle filter: multitarget detection and tracking. *European Conference on Computer Vision*, 1:28–39, 2004.

[50] Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, February 2002.

[51] Rui, Y. and Chen, Y. Better proposal distributions: Object tracking using unscented particle filter. In *CVPR (2)*, pages 786–793. IEEE Computer Society, 2001.

[52] Djurić, P. M. et al. Applications of particle filtering to selected problems in communications: a review and new developments. *IEEE Signal Processing Magazine, September, 2003*, September 2003.

[53] Doucet, A., Godsill, S., and Andrieu, C. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[54] Isard, M. and Blake, A. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[55] Comaniciu, D. and Meer, P. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[56] Comaniciu, D., Ramesh, V., and Meer, P. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[57] Collins, R. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition (CVPR'03)*. IEEE, June 2003.

[58] Bretzner, L. and Lindeberg, T. Qualitative multi-scale feature hierarchies for object tracking. *Journal of Visual Communication and Image Representation*, 11(2):115–129, 2000.

[59] Lindeberg, T. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[60] Lepetit, V. and Fua, P. *Monocular Model-Based 3-D Tracking of Rigid Objects.* Now Publishers Inc, 2005.

[61] Tomasi, C. and Kanade, T. Detection and tracking of point features. *School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91-132*, 1991.

[62] Birchfield, S. Derivation of Kanade-Lucas-Tomasi tracking equation. *Unpublished, May*, 1996.

[63] Shi, J. and Tomasi, C. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[64] Jin, H., Favaro, P., and Soatto, S. Real-time feature tracking and outlier rejection with changes in illumination. *Technical report*, 2000.

[65] Sinha, S., Frahm, J. M., and Pollefeys, M. GPU-based video feature tracking and matching. *Workshop on Edge Computing Using New Commodity Architectures*, 2006.

[66] Illingworth, J. and Kittler, J. A survey of the Hough transform. *Comput. Vision Graph. Image Process.*, 44(1):87–116, 1988.

[67] Leavers, V. F. Which Hough transform? *CVGIP: Image Underst.*, 58(2):250–264, 1993.

[68] Karabernou, S. M., Kessal, L., and Terranti, F. Erratum: Erratum to" Real-time FPGA implementation of Hough transform using gradient and CORDIC algorithm"[Image and Vision Computing 23 (2005) 1009-1017]. *Image and Vision Computing*, 25(6):1032, 2007.

[69] van Ginkel, M., Hendriks, C. L. L., and van Vliet, L. J. A short introduction to the Radon and Hough transforms and how they relate to each other, 2004.

[70] Ballard, D. H. Generalizing the Hough transform to detect arbitrary shapes. In Fischler, M. A. and Firschein, O., editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 714–725. Kaufmann, Los Altos, CA., 1987.

[71] Stockman, G. C. and Agrawala, A. K. Equivalence of Hough curve detection to template matching. *Commun. ACM*, 20(11):820–822, 1977.

[72] J. Princen, J. Illingworth, J. K. A formal definition of the Hough transform: properties and relationships. *Journal of Mathematical Imaging and Vision*, 1:153 – 168, Jul 1992.

[73] Merlin, P. M. and Farber, D. J. A parallel mechanism for detecting curves in pictures. *IEEE Trans. Computers*, 24(1):96–98, 1975.

[74] Chau, C.-P. and Siu, W.-C. Generalized Hough transform using regions with homogeneous color. *Int. J. Comput. Vision*, 59(2):183–199, 2004.

[75] Fung, P. F., Lee, W. S., and King, I. Randomized generalized Hough transform for 2-D grayscale object detection. *Proceedings of the 13th International Conference on Pattern Recognition*, 2:511–515, 1996.

[76] Kälviäinen, H. *Randomized Hough Transform: New Extensions.* Lappeenranta University of Technology, 1994.

[77] Gerig, G. and Klein, F. Fast contour identification through efficient Hough transform and simplified interpretation strategy. *Proceedings 8th International Conference on Pattern Recognition, Paris, France, Oct*, pages 27–31, 1986.

[78] Gerig, G. Linking image-space and accumulator-space: a new approach for object recognition. *Proceedings First International Conference on Computer Vision (ICCV'87), Computer Society of the IEEE and International Association for Pattern Recognition (IAPR), London, England, June*, pages 8–11, 1987.

[79] Li, H., Lavin, M. A., and Le Master, R. J. Fast Hough transform: a hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 36(2-3):139–161, 1986.

[80] Kiryati, N., Kaelviaeinen, H., and Alaoutinen, S. Randomized or probabilistic Hough transform: unified performance evaluation. *Pattern Recognition Letters*, 21(13-14):1157–1164, 2000.

[81] Duda, R. O. and Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[82] Bray, M., Koller-Meier, E., and Van Gool, L. Smart particle filtering for high-dimensional tracking. *Computer Vision and Image Understanding*, 106(1):116–129, 2007.

[83] Jain, R., Kasturi, R., and Schunck, B. G. *Machine Vision.* McGraw-Hill, Inc., 1995.

[84] Low, A. *Introductory Computer Vision and Image Processing.* Mcgraw Hill Book Co Ltd, 1991.

[85] Smith, S. M. and Brady, J. M. Susan-a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, 1997.

[86] Alexandrescu, A. *Modern C++ Design: Generic Programming and Design Patterns Applied.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[87] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[88] Kiczales, G. et al. Aspect-oriented programming. In Akşit, M. and Matsuoka, S., editors, *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, Berlin, Heidelberg, and New York, 1997.

[89] Veldhuizen, T. Using C++ template metaprograms. *C++ Report*, 7(4):36–43, may 1995.

[90] The Boost MPL library. [online, last accessed October 2007]. http://www.boost.org/libs/mpl/doc/.

[91] Köthe, U. Reusable software in computer vision. In B. Jähne, H. Haussecker, P. G., editor, *Handbook of Computer Vision and Applications, Volume 3: Systems and Applications*, pages 103–132. Academic Press, 1999.

[92] Köthe, U. STL-style generic programming with images. *C++ Report Magazine*, 12(1):24–30, January 2000.

[93] Boissenin, M., Wedekind, J., Amavasai, B., Caparrelli, F., and Travis, J. Fast pose estimation for microscope images using stencils. IEEE Systems, Man and Cybernetics Society, 2006.

[94] Isard, M. and Blake, A. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[95] Goldenshluger, A. and Zeevi, A. The Hough transform estimator. In *Annals of statistics*, volume 32, pages 1908–1932, 2004.

[96] Rousseeuw, P. J. and Leroy, A. M. *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc., 1987.

[97] MMVL/SHU. Mimas, open source computer vision library. MMVL wiki pages [online], 2008. http://www.shu.ac.uk/mmvl/mimas/.

[98] Wedekind, J. and Boissenin, M. Micron vision package, 2006. http://vision.eng.shu.ac.uk/mediawiki/index.php/MiCRoN_Microscope_Vision_Software.

[99] Attneave, F. and Arnoult, M. The quantitative study of shape and pattern perception. *Psychological Bulletin*, 53(6):452–471, 1956.

[100] Vacchetti, L., Lepetit, V., and Fua, P. Stable real-time 3-D tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, 2004.

[101] Optometrists network. http://www.vision3d.com/stereo.html, last accessed October 2007.

[102] Project - the self-reconfigurable camera array. [online, last accessed October 2007]. http://amp.ece.cmu.edu/projects/MobileCamArray/.

[103] Roy, S. and Cox, I. J. A maximum-flow formulation of the n-camera stereo correspondence problem. In *ICCV*, pages 492–502, 1998.

[104] Kolmogorov, V., Zabih, R., and Gortler, S. J. Generalized multi-camera scene reconstruction using graph cuts. In Rangarajan, A., Figueiredo, M. A. T., and Zerubia, J., editors, *EMMCVPR*, volume 2683 of *Lecture Notes in Computer Science*, pages 501–516. Springer, 2003.

[105] Kolmogorov, V. and Zabih, R. Computing visual correspondence with occlusions using graph cuts. *International Conference on Computer Vision*, pages 508–515, 2001.

[106] Boykov, Y. and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[107] Blake, A. Markov random fields, graph cut optimization and applications to machine vision, may 2007.

[108] Kolmogorov, V. and Zabih, R. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[109] Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., and Rother, C. Bi-layer segmentation of binocular stereo video. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2, 2005.

[110] Rother, C., Kolmogorov, V., and Blake, A. " GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[111] Rother, C., Kumar, S., Kolmogorov, V., and Blake, A. Digital tapestry. *Proc. Conf. Comp. Vision and Pattern Recog*, 2005.

[112] Scharstein, D., Szeliski, R., and Zabih, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, 2001.

[113] Oggier, T. et al. An all-solid-state optical range camera for 3-D real-time imaging with sub-centimeter depth resolution (SwissRanger). *Proc. SPIE*, 5249:534–545, 2004.

[114] Johnson, A. E. *Spin-Images: A Representation for 3-D Surface Matching.* PhD thesis, Carnegie Mellon University, March 1997.

[115] Lowe, D. G. Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2, 1999.

[116] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[117] Mikolajczyk, K. and Schmid, C. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[118] Ke, Y. and Sukthankar, R. PCA-SIFT: A more distinctive representation for local image descriptors. *Proc. CVPR*, 2:506–513, 2004.

[119] Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded Up Robust Features. *Lecture Notes in Computer Science*, 3951:404, 2006.

[120] Mikolajczyk, K., Leibe, B., and Schiele, B. Local features for object class recognition. *Proc. ICCV*, 2:1792–1799, 2005.

[121] Perona, P. An invitation to visual recognition. Vmath, January 2005. [video],last visited october 2007.

[122] Perona, P. An exploration of visual recognition. The internet archive, March 2007. [video].

[123] Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[124] Faucher, J. Camera calibration and 3-D reconstruction. Master's thesis, ENSEIRB, 2006.

[125] Chang, M. M. Y. and Wong, K. H. Model reconstruction and pose acquisition using extended Lowe's method. *Multimedia, IEEE Transactions on*, 7(2):253–260, 2005.

[126] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. Bundle adjustment–A modern synthesis. *Vision Algorithms: Theory and Practice*, 1883:298–372, 2000.

[127] Lowe, D. G. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[128] Araujo, H., Carceroni, R. L., and Brown, C. M. A fully projective formulation to improve the accuracy of lowe's pose estimation algorithm. *Coordinates*, 10:0.

[129] Araujo, H., Carceroni, R., and Brown, C. A fully projective formulation to improve the accuracy of Lowe's pose-estimation algorithm, 1998.

[130] Wong, A. K. C., Rong, L., and Liang, X. Robotic vision: 3-D object recognition and pose determination. *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, 2, 1998.

[131] Mühlmann, K., Maier, D., Hesser, J., and Männer, R. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1-3):79–88, 2002.

[132] Brown, M., Drummond, T., and Cipolla, R. 3-D model acquisition by tracking 2-D wireframes. *Electronic Proceedings of The Eleventh British Machine Vision Conference University of Bristol*, pages 11–14, 2000.

[133] Hill, J.-R. Linux goes 3-D: An introduction to Mesa/Opengl. [online, last accessed October 2007], 1996. http://www.linuxjournal.com/article/0174.

[134] Galimberti, R. An algorithm for hidden line elimination. *Commun. ACM*, 12(4):206–211, 1969.

[135] Woo, M., Neider, J., and Davis, T. *OpenGL Programming Guide (Second Edition).* Addison Wesley, 1997. ISBN 0-201-46138-2.

[136] Sutherland, I. E., Sproull, R. F., and Schumacker, R. A. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, 1974.

[137] Batchelor, B. G. and Whelan, P. F. Machine vision systems: Proverbs, principles, prejudices and priorities,". *Machine Vision Applications, Architectures, and Systems Integration*, 3:374–383.