

Three-dimensional motion capture for the DIET breast cancer imaging system

Richard Brown

A thesis presented for the degree of
Doctor of Philosophy
in
Mechanical Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

June 2008

Soli Deo Gloria

ACKNOWLEDGEMENTS

The production of this thesis has been a challenging, rewarding, and an enjoyable experience. This is due in no small part to a number of individuals whom I would like to acknowledge and thank.

Thanks to my supervisors, Geoff Chase and Chris Hann and to the researchers and students who have been or are still associated with the DIET group: Eli van Houten, Ashton Peters, Rodney Elliot, Hans Uwe-Berger, Benjamin Petit, Micheal Wiertlewski, Crispin Berg, Anthony Hii, Shig Kinoshita, Geoff Rodgers, Edouard Ravni, and Fabrice Jandet. It's been great being part of the team and sharing in the exchange of ideas.

Thank you Emily, for encouraging me to do this, and for your unwavering love and support. You are the best wife I could ever have hoped for and I couldn't have done it without you. Sam and Tom, when you're old enough to read this, you've been a source of great joy, encouragement and refreshing to your Daddy, even when he's had to spend long hours working, which I know you haven't really understood. I am blessed with a wonderful family.

Thank you to my extended family for all of your help, encouragement, prayers, and understanding¹.

Finally, I would like to acknowledge the support of the Tertiary Education Commission with funding from a Top Achiever Doctoral scholarship over the three years of this thesis.

¹Kate, I couldn't work out how to incorporate the word *hippopotamus* into this document, sorry

CONTENTS

1	ABSTRACT	xvii
2	INTRODUCTION	1
2.1	Digital Image-based ElastoTomography (DIET)	2
2.2	Thesis scope and preface	3
2.2.1	Image capture	4
2.2.2	Camera calibration	4
2.2.3	Feature tracking	5
2.2.4	3D reconstruction	5
2.3	Thesis organisation	6
3	BACKGROUND	7
3.1	Notation	7
3.2	Projective geometry of the plane	7
3.2.1	Homogeneous representation of points and lines	7
3.2.2	Intersection of points and lines	8
3.2.3	A model for the projective plane	9
3.2.4	Conics in the projective plane	9
3.2.5	Projective transformations	9
3.2.5.1	Computing a homography from four point correspondences	10
3.2.5.2	Projective transformation of lines	10
3.2.5.3	Projective transformation of conics	10
3.2.5.4	The cross ratio	10
3.3	Projective Space	11
3.4	Single view geometry	12
3.4.1	Camera Model	12
3.5	Calibration	14
3.5.1	Camera resection from point correspondences	14
3.5.1.1	Algebraic error minimisation	16
3.5.1.2	Geometric error minimisation	16
3.5.1.3	Normalisation	17
3.5.2	Factorising the projection matrix	18

3.6	Multiple view geometry	18
3.6.1	Epipolar geometry	18
3.6.2	The Fundamental matrix	19
3.6.3	Triangulation	20
3.7	Summary	21
4	CAMERA CALIBRATION: INTRODUCTION AND INITIAL APPROACH	23
4.1	Introduction	23
4.1.1	Close-range camera calibration with a rig	24
4.1.2	DIET requirements	24
4.1.3	Calibration rig design	25
4.2	Calibration using the grid-based cube	28
4.2.1	Feature identification	28
4.2.1.1	Line detection	28
4.2.1.2	Finding sets of parallel lines by robust vanishing point detection	32
4.2.1.3	Vertex detection	32
4.2.2	Feature correspondence	37
4.3	Experimental Results	38
4.4	Discussion and Conclusion	41
5	SILHOUETTE-BASED CALIBRATION	43
5.1	Introduction and Overview	43
5.2	Method	44
5.2.1	Algorithm overview	44
5.2.2	Identifying the cube silhouette and elliptical features	45
5.2.3	Partitioning the image into cube faces	47
5.2.3.1	Partitioning the cube into two faces	47
5.2.3.2	Partitioning the cube into three faces	49
5.2.3.3	Degenerate viewing angles	50
5.2.4	Finding point correspondences	50
5.2.4.1	Mapping image faces to the unit square	51
5.2.4.2	Mapping cube faces into the unit square	53
5.2.4.3	Matching image and world features	55
5.2.5	Finding image coordinates of circle centres	57
5.2.6	Camera resection	57
5.3	Case study	57
5.4	Discussion and Conclusion	60
6	FEATURE TRACKING	63
6.1	Introduction	63
6.2	Nearest neighbour matching: small motion	65
6.3	Invariant signatures	66
6.3.1	Euclidean-invariant signature for ordered point sets	66

6.3.2	Similarity-invariant signature for ordered point sets	68
6.4	Point tracking with a 3-point Euclidean invariant signature	69
6.4.1	3-point ordered similarity invariant	71
6.5	A similarity-invariant approach without ordered points	71
6.6	Proof of concept experiments	75
6.6.1	Computer simulation	75
6.6.1.1	Motion model	75
6.6.1.2	Validation of Euclidean assumption	76
6.6.1.3	Experiments	78
6.6.1.4	Results	79
6.6.2	Gel phantom experiment	81
6.6.2.1	Experiment	81
6.6.2.2	Summary	84
6.7	Summary	84
7	SEER: A 3D SURFACE RECONSTRUCTION ALGORITHM	85
7.1	Introduction and prior work	85
7.1.1	Development Approach	87
7.2	3D Reconstruction Algorithm	88
7.2.1	Epipolar point cloud construction	88
7.2.2	Parameters	89
7.2.3	RANSAC tangent plane estimation	90
7.2.4	Reconstruction algorithm	91
7.2.5	Implementation Notes	92
7.3	Case study	94
7.3.1	Computer Simulation	94
7.3.1.1	Performance under varying point density	96
7.3.1.2	Effect of parameters $n_B, \kappa, \rho_{\text{est}}$	97
7.4	Discussion and Conclusion	97
8	EXPERIMENTAL DIET SYSTEM	99
8.1	Introduction	99
8.2	System components	100
8.2.1	Overview	100
8.2.2	Hardware platform	100
8.2.3	Actuator	102
8.2.4	Cameras	102
8.2.5	Strobe: LED ring flashes	103
8.3	Control software	103
8.3.1	dSpace controller	103
8.3.1.1	Actuator amplitude control	104
8.3.1.2	Strobe signal generation	105
8.3.2	Camera Server	106
8.3.2.1	Camera initialisation and calibration procedure	106
8.3.2.2	Experiment procedure	106

8.4	Results	106
8.5	Labview	107
8.6	Summary	108
9	CASE STUDY	111
9.1	Introduction	111
9.2	Methodology	111
9.2.1	Gel phantom	111
9.2.2	Camera calibration	113
9.2.3	Fiducials	113
9.2.4	Image capture	113
9.2.5	Feature extraction	113
9.2.5.1	Error analysis	116
9.2.6	Tracking	116
9.2.7	3D surface reconstruction	117
9.2.8	3D motion reconstruction	119
9.3	Experiments	121
9.3.1	Five cameras	121
9.3.2	Various sized inclusions	121
9.3.2.1	One possible approach to detecting breast inhomogeneity from 3D ellipse characteristics	122
9.4	Summary	122
10	CONCLUSIONS	127
11	FUTURE WORK	131
11.1	DIET	131
11.2	Camera calibration	132
11.3	Signature tracking	132
11.4	SEER	132

NOMENCLATURE

Mathematics

\mathbb{R}^n	n -dimensional vector space over the real numbers
\mathbb{P}^n	n -dimensional projective space, with coordinates in \mathbb{R}^{n+1}
$\mathbb{R}^{m \times n}$	The space of $m \times n$ real-valued matrices
A	Typewriter type signifies a matrix
\mathbf{A}^\top	The transpose of A
$\mathbf{A}^{-\top}$	$(\mathbf{A}^{-1})^\top$ or $(\mathbf{A}^\top)^{-1}$ (they are equivalent)
x	Bold face type signifies a column vector $(x_1, x_2, \dots, x_n)^\top$
$\ \mathbf{x}\ $	Euclidean norm of x , i.e. $(\sum_i x_i^2)^{\frac{1}{2}}$
$\arg \min_x f(x)$	The value of x that minimises $f(x)$

Additional notation will be introduced when used.

Camera model and geometry

K	Intrinsic camera calibration matrix $\in \mathbb{R}^{3 \times 3}$
R	Rotation matrix $\in \mathbb{R}^{3 \times 3}$, with $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ and $ R = 1$
P	Camera projection matrix $\in \mathbb{R}^{3 \times 4}$
H	Matrix representation of a homography in homogeneous coordinates
Q	Matrix representation of a conic

Acronyms

ACD	Analogue to Digital Converter
CCD	Charge-Coupled Device
CNC	Computer Numerically Controlled
DIET	Digital Image-based ElastoTomography
DLT	Direct Linear Transformation
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
LED	Light-Emitting Diode
LVDT	Linear Variable Differential Transformer
MLE	Maximum Likelihood Estimate
MMRANSAC	Multiple-Model RANSAC
MR	Magnetic Resonance
NCC	Normalised Cross-Correlation
PID	Proportional Integral Derivative
RANSAC	RANdom Sample And Concensus
SDK	Software Development Kit
SEER	Surface Extraction from Epipolar-constrained Reconstruction
SIFT	Scale-Invariant Feature Transform
SUSAN	Smallest Univalued Segment Assimilating Nucleus
SVD	Singular Value Decomposition
TCP/IP	Transmission Control Protocol / Internet Protocol
TPS-RPM	Thin-Plate Spline Robust Point Matching

LIST OF FIGURES

3.1	The projective geometry of concurrent lines	11
3.2	Pinhole camera geometry. \mathbf{C} is the camera centre, \mathbf{p} is the principal point. The 3D coordinate system is placed with the origin at \mathbf{C} and the Z axis aligned with the optical axis of the camera. The retinal plane is the plane $Z = f$ where f is the focal length of the camera. A point \mathbf{X} projects to a point \mathbf{x} in the image by intersecting the line joining \mathbf{C} and \mathbf{X} with the plane	13
3.3	Epipolar constraint	18
3.4	Epipolar geometry: The intersection of the epipolar plane constructed from the baseline and \mathbf{X} with the two image planes gives two epipolar lines \mathbf{l}, \mathbf{l}' . Potential matches in the second image to \mathbf{x} must fall on \mathbf{l}' , and vice versa.	19
4.1	First calibration cube, on a white background	26
4.2	Second calibration cube	27
4.3	Third calibration cube	27
4.4	Edges of the first calibration cube of Fig. 4.1 found by the Sobel edge detector	30
4.5	Close up of a peak from the Hough and Radon transforms of the edge image in Fig. 4.4 for essentially the same parameter space. Note how the Radon image is smoother, and the peak more easily localised	30
4.6	Line parametrisation used for the Radon transform. The image line \mathbf{l} (shown in bold) is parametrised by (ρ, θ) where ρ is the perpendicular distance from \mathbf{l} to the centre of the image, and θ the clockwise angle from the horizontal to the corresponding normal to \mathbf{l}	31
4.7	All detected Radon transform peaks, and the corresponding image lines	33

4.8	Detected vanishing points and inliers from MMRANSAC algorithm for vanishing point estimation	35
4.9	Detected vertices, organised by face	38
4.10	Calibration user interface	39
4.11	The two views of the cube used to calibrate the two cameras	40
4.12	Computed position and orientation of the two cameras, with respect to the cube	41
4.13	Reconstructed features from two views (blue dots - the cube boundary is shown for context)	42
5.1	Two different views of the calibration cube used to illustrate the calibration procedure. Note that the cube is easy to segment from the background but that the interior edges are indistinct and therefore difficult to determine by standard image processing techniques	45
5.2	Identifying the silhouette and elliptical features for the cube depicted in Fig. 5.1b	46
5.3	Partitioning the cube into two faces for the two faces visible case	48
5.4	The vanishing point v_1 is constructed by intersecting the lines containing \overline{ab} and \overline{de} . The closer of the two remaining vertices (c and f) to v_1 is vertex c , so the line passing through v_1 and c will pass along one of the cube edges, through the centre vertex. The same procedure is followed to construct v_2 and v_3 and the other two lines passing through the centre. The centre, g , is computed as the intersection of the three lines by least squares. The boundaries of the three faces of the cube are therefore (g, a, b, c) , (g, c, d, e) , (g, e, f, a) , ordered anticlockwise from g by convention	49
5.5	Degenerate viewing angles	50
5.6	Mapping face ‘4’, including the ellipses, from Fig. 5.1b to the unit square. Note that due to small inaccuracies in determining the ellipses and the corners of the face, the projective distortion is not completely removed, the features are still slightly elliptical after transformation.	53
5.7	Embedding the world cube into \mathbb{R}^3	54
5.8	The world coordinates of circle centres and corners of the ‘4’ face, mapped to the unit square	55
5.9	The four possible orientations of the ‘4’ face mapped from the image. Each of these four rotations of the ‘4’ face match the world ‘4’ face depicted in Fig. 5.8	56

5.10	Five views of the white calibration cube, together with their fitted elliptical features and face boundaries	58
5.11	Reconstructed camera locations and orientations with respect to the cube	59
6.1	Point motion where the motion is significantly less than the point spacing	64
6.2	Point motion where the motion is larger than the point spacing	65
6.3	Constructing the 3-point Euclidean invariant	70
6.4	Finite element gel cylinder model used for simulation, showing one frame of the cylinder in steady state motion	75
6.5	Points on the cylinder as measured from the camera	77
6.6	Image point trajectories for different levels of measurement noise. The noise is Gaussian with standard deviation σ in each coordinate	77
6.7	The motion of each of the feature points	78
6.8	Gel cylinder used in tracking experiment	82
6.9	Trajectories of successfully tracked points in gel phantom experiment	83
7.1	Views of a phantom from two different cameras, shown together with a small neighbourhood of each red point	86
7.2	Epipolar constraint and point cloud formation. Any choice of two points on \mathbf{l} and \mathbf{l}' will satisfy the epipolar constraint (§3.6.1). In this image, the two world points $\mathbf{X}_1, \mathbf{X}_2$ generate two image points in each image. Each pair of points $(\mathbf{x}_1, \mathbf{x}'_1), (\mathbf{x}_1, \mathbf{x}'_2), (\mathbf{x}_2, \mathbf{x}'_1), (\mathbf{x}_2, \mathbf{x}'_2)$ satisfies the epipolar constraint, and the resulting constructed points in space are the true surface points $\mathbf{X}_1, \mathbf{X}_2$, together with two additional points \mathbf{U}, \mathbf{V} resulting from incorrect correspondences.	89
7.3	Constraints for the plane fitting procedure	93
7.4	Epipolar point clouds where one or three colours are used	94
7.5	Surface reconstruction process for a 5cm radius hemisphere with points randomly distributed with a mean density of $\rho = 10$ pts/cm ²	95
8.1	System diagram of the experimental setup showing the major components	101
8.2	Actuator with gel phantom. Four of the five cameras, mounted with LED ring flashes can also be seen	102
8.3	Control Desk GUI used for observation and parameter adjustment. The plot shows the LVDT measurement and the strobe pulse	104

8.4	Actuator amplitude controller	105
8.5	Computation of the strobe pulse train from thresholded LVDT signal $L(t)$	105
8.6	Sequence diagram for capturing a data set. Note, this illustrates a simple example where only 3 images are required for each camera.	107
8.7	LVDT and strobe signals for different phase increments for a signal of period T . Note, the display is triggered by the strobe signal, and hence it appears that the strobe signal is stationary while the LVDT signal moves, when in reality it is the other way around	109
8.8	Still image of actuated gel phantom frozen by strobing. Note that the image is crisp and in focus	110
9.1	The gel phantom used. The phantom is suspended from an aluminium plate, and fits snugly in, but is not supported by, an aluminium interface screwed to the actuator plate	112
9.2	The 20 images obtained from one camera	114
9.3	Feature extraction procedure	115
9.4	Errors in the measured centroids. The red lines show the Gaussian curves constructed from the computed means and standard deviations, showing that the error can be considered to be Gaussian	117
9.5	Tracked features from the image sequence in Fig. 9.2	118
9.6	A band of epipolar lines generated by Monte carlo simulation. The width of the band is approximately 4 pixels, so a matching threshold of 2 pixels is chosen	119
9.7	Reconstructed 3D points for the four-camera experiment. Note the gaps in the sides of the reconstruction due to insufficient cameras	120
9.8	Reconstructed 3D motion of each point	120
9.9	Reconstructed 3D points when five cameras are used. Note the gaps in the sides of the reconstruction from Fig. 9.7 are no longer present, as five cameras are sufficient	121
9.10	Motion of the phantom with large inclusion	123
9.11	Measured out of plane component η for all 3D trajectories for phantoms with no inclusion, and small, medium, and large inclusions respectively	124
11.1	Possible clinical experimental setup	131

LIST OF TABLES

4.1	Computed 3D coordinates of the optical centre and the intrinsic calibration matrix for each camera	40
5.1	Image coordinates and desired unit square coordinates of the corners of the ‘4’ face of the cube depicted in Fig. 5.1b. Note, the coordinates are homogeneous coordinates	52
5.2	Mappings into the unit square from each face of the cube	53
5.3	Reconstructed 3D coordinates of the optical centre and the intrinsic calibration matrix for each camera	60
5.4	Reconstruction results for pairs of adjacent cameras	60
6.1	Results for $\sigma = 0$	79
6.2	Results for $\sigma = 0.2$	80
6.3	Results for $\sigma = 0.5$	80
7.1	Surface reconstruction for varying point density ρ . The last two rows have $\kappa = 0.2$, the rest have $\kappa = 0.1$	96
7.2	Results for computer simulation varying $n_B, \kappa, \rho_{\text{est}}$	98

Chapter 1

ABSTRACT

Breast cancer is one of the most prevalent forms of cancer in the world today. The search for effective treatment and screening methods is a highly active area of research. The Digital Image-based ElastoTomography (DIET) project is a new breast cancer screening system under development, where surface motion from the mechanically actuated breast is measured in 3D, and used as input to an inverse problem solving for breast elasticity. Cancerous lesions appear as high contrast features, being an order of magnitude stiffer than healthy tissue. The 3D motion capture is measured by an array of digital cameras using computer vision techniques.

This thesis presents a complete imaging system and algorithms for the capture of 3D breast surface motion. The main components of the 3D motion capture system are the hardware and software image capture system, camera calibration, intra-image feature tracking, and 3D surface and motion reconstruction. Accurate algorithms for each of these components are developed, with a view to future development and potential modifications needed for a clinically-appropriate system. A number of the algorithms developed have potential applications outside of the DIET system.

Proof of concept studies demonstrate the viability of the system, with full motion reconstruction being performed on silicone gel phantoms, designed to approximate human soft tissue, in a number of laboratory experiments.

Chapter 2

INTRODUCTION

It has long been a dream of scientists and engineers to equip machines with the ability to see. Sight is an ability that we humans take for granted. We can instantly recognise objects, faces, and colours from the slightest of cues without pause for thought. We can look at an object and know where it is relative to us. We can see a car moving in the distance and work out which direction it is headed, as well as estimating its speed surprisingly accurately. Seeing a small hard ball travelling towards at over 100 kmph we can extend one hand and pluck it out of the air.

The complexity of all of these tasks is perhaps not apparent until one attempts to replicate them on a machine. Consider for instance the ubiquitous CAPTCHA¹. They can be found on many websites, where a user is prompted to enter the sequence of characters that appear in a distorted image before they may proceed. It almost seems absurd that it is very difficult to recognise the sequence of characters automatically with a computer, but this is the CAPTCHA's very *raison d'être*, with many current uses that seek to prevent automated logon or entry by a computer.

It was recognised long ago that the primary reason that humans can perceive depth, particularly of near objects, is because two slightly offset views of the scene are received from two eyes. The brain matches the two stereoscopic views received and enables perception of 3D structure from the disparity between the two views. It was realised in the early 20th century that it is possible, given two views of a scene from two cameras, to reconstruct the 3D structure of the scene, given some additional information. First, the relative position and orientation of the two cameras is required, together with their focal length. If these properties are known, the two cameras are calibrated, and it is possible to compute the 3D position of a point from its coordinates in the two images. For modern digital CCD² cameras, some additional parameters are required, in particular the mapping from the retinal plane (imaging surface) to pixel coordinates.

¹Completely Automated Public Turing test to tell Computers and Humans Apart

²Charge-Coupled Device

With the advent of fast computers capable of processing large amounts of image data quickly, and inexpensive, high-quality digital cameras, the field of computer vision has rapidly matured into a major research field. Computer vision has created many new applications and simplified many existing ones. Examples of computer vision systems are 3D terrain measurement from aircraft, manufacturing quality control, autonomous navigation of cars and aeroplanes, computer graphics, augmented reality devices, large-scale 3D urban scenery measurement, and motion analysis for sport and physiotherapy.

2.1 Digital Image-based ElastoTomography (DIET)

It is well known that breast cancer is one of the most prevalent and harmful diseases among women in modern society. For example, in New Zealand it was the leading form of cancer death in women in 1999 [28]. For a high chance of successful treatment, it is critical that malignant tumours are detected as early as possible. If a carcinoma is detected sufficiently early, the chances of five-year survival increase to approximately 90% [1].

The current standard breast cancer screening method is mammography, which is currently used in wide-scale screening programs in most developed nations. Mammography measures variation in tissue radio-density. However, the contrast between healthy and diseased tissue is relatively low (5% - 10%), which creates difficulties with patients with higher density breast tissue [37], including most younger women.

The search for effective imaging methods thus remains an active area of research, and elastographic techniques are beginning to play a significant role. Instead of imaging tissue radio-density, elastography measures tissue stiffness. Breast tissue stiffness is a high-contrast feature with an order of magnitude variation or more between normal and cancerous tissue [21]. Hence, it is potentially a much better feature to image. Emerging elastographic breast cancer screening techniques include mechanical sensors on the skin surface [34], ultrasound elastography [5], and MR elastography [40].

A Digital Image-based ElastoTomography (DIET) system is being developed at the University of Canterbury, for measuring breast elasticity by elastotomography. The breast is mechanically actuated by a low frequency ($\approx 50 - 100\text{Hz}$), low amplitude ($\approx 1\text{mm}$) sinusoid. This motion induces a periodic steady-state motion throughout the breast. Motion at the breast surface can be used to solve an inverse problem for the elasticity distribution throughout the breast [32]. More specifically, it is assumed that under these conditions the breast obeys a damped linear elasticity model, and therefore that the motion of an individual point on the surface is harmonic, following an elliptical trajectory. The motion of individual points can thus be represented in a 3D Cartesian coordinate system by an amplitude and a phase offset in each coordinate. It is these amplitude and phase measurements, of a reasonably dense set of surface

points, that is used as the input to the inverse problem. The solution to the inverse problem is an ongoing area of research. Using a combination of finite element analysis and optimisation techniques, it is now possible to locate a tumour of known size, and estimate its elastic properties in silicon gel phantoms used in laboratory experiments [32]. The full inverse problem of estimating the elasticity throughout the breast volume is still being investigated.

The DIET system requires a means of accurately measuring the surface motion of densely distributed points or features. There are not many viable techniques for measuring motion at this relatively high density. It is impractical to mount mechanical sensors on the surface, such as accelerometers, because the motion of a large quantity of features is required. It would theoretically be possible to use a custom built laser scanner to measure point motion, but this approach would be very costly, require a significant amount of hardware development, and require multiple lasers and sensors to measure the 3D motion. There is thus one imaging modality that clearly stands out as the ideal choice of imaging system: computer vision. A computer vision system would theoretically allow the 3D reconstruction of any portion of the breast surface from images from two calibrated cameras. The advantages of computer vision over its alternatives are manifold: The equipment is inexpensive, the 3D reconstruction can be performed to an almost arbitrarily high resolution, and the equipment is portable.

This thesis presents a computer vision system and algorithms for the measurement of 3D breast surface motion, ranging from the image capture hardware system through to the reconstruction of the 3D motion of features on the breast surface.

2.2 Thesis scope and preface

This thesis addresses all aspects of the imaging component of the DIET system. The major components of the thesis are:

Image capture A hardware system for automatically capturing digital images in a laboratory context, including actuator control. The image capture stage includes image processing methods for extracting point features from images

Camera calibration Methods for easy and accurate calibration of an array of digital cameras, based on the design of a custom calibration object

Feature tracking A means of tracking the motion of individual point features between frames from a single camera

3D Reconstruction An algorithm for reconstructing a 3D surface from images from each of the cameras

Each of these components is described in more detail in the following subsections.

2.2.1 Image capture

To measure the 3D trajectory of a point, it is first necessary to know the 3D position of that point at a number of closely-spaced instances of time. For the DIET system, given that the actuation frequency is of the order of 100Hz, this task involves taking images that are temporally separated by less than 1ms. Standard video cameras typically capture between 20 and 30 images per second, which is an order of magnitude too slow. High speed cameras capable of the required frame rate are available, but are prohibitively expensive and have relatively low resolution compared to still cameras.

Given that the breast is harmonically (sinusoidally) actuated, when the system is in a steady state the motion of the surface will be periodic at the actuation frequency. Therefore, the surface can be made to appear static by illuminating the surface with a strobe that provides a short burst of intense light once per cycle at a fixed phase offset from the actuator. An image can then be taken of the surface, without concern for the shutter speed. Hence, a single image may be built up from hundreds of pulses of light from the strobe. Images at different points of the actuator cycle can then be taken by varying the phase offset of the strobe.

The system presented in Chapter 8 is a complete image capture system, comprising synchronised control of the actuator, cameras, and strobe light and allowing fully automatic capture of entire image sequences.

2.2.2 Camera calibration

A critical component of any computer vision system involving accurate 3D reconstruction is camera calibration. Camera calibration provides a mathematical model for the action of the camera. More specifically, it creates or identifies a model describing how points in 3D are mapped to 2D image coordinates. If cameras are inaccurately calibrated, then the 3D reconstruction will be inaccurate, or not even possible to compute.

Typically, cameras are calibrated by taking images of an object with accurately known dimensions, and computing a model that maps features on the object to their image locations. The algorithms presented in Chapters 4 and 5 are based on this approach. A calibration cube had been produced for preliminary DIET investigations prior to the commencement of this research. However, the method used for calibrating the camera from images of the cube was unreliable, and required a large amount of user input, including manual specification and correspondence of image feature points, and image enhancement in image editing software such as Photoshop. The algorithm presented in Chapter 4 was designed to reliably calibrate cameras from images of this existing calibration cube, almost fully automatically. The resulting algorithm was successful for experiments involving two cameras, but when the system was expanded to incorporate four or more cameras, calibration became almost prohibitively difficult.

The algorithm only worked for an overly restrictive range of camera viewpoints, and the time taken to calibrate the cameras was very long, at around two minutes per camera.

As a result, Chapter 5 presents a new calibration algorithm designed to replace the existing one, together with a new calibration cube design. The new calibration algorithm allows fully automatic camera calibration from a single image. It also enables a much wider range of possible camera viewpoints, and takes only ≈ 4 seconds per camera³. The same levels of accuracy are maintained or even improved, despite using lower resolution images.

2.2.3 Feature tracking

To compute the 3D trajectories of individual points, the points must first be tracked between each frame within each individual camera. Rather than tracking by correlating image intensity regions, point features are tracked geometrically. Depending on the density of the point features and the magnitude of the motion, this task can be difficult because the motion of the point features may be larger than the interpoint spacing. The approach taken in Chapter 6 is to treat the transformation between frames as being able to be locally approximated by transformations from a transformation group. Euclidean and similarity transformations are considered. Local joint invariants of the transformation group are used to match a set of key points, and the remaining points are matched by interpolating the motion of the key points. The key point matching relies on the random distribution of the points. If the motion of the points is small compared with the point spacing, geometric features can be simply tracked using nearest neighbour matching, which is a very fast procedure.

Both the feature tracking and 3D reconstruction algorithms developed in this thesis use randomly distributed image point features in three colours as geometric primitives. A system of fiducials (markers) is presented that are easily segmented from the images and identified based on colour. The fiducial system and image extraction technique are presented in Chapter 9.

2.2.4 3D reconstruction

An algorithm, Surface Extraction from Epipolar-constrained Reconstruction (SEER), is developed in Chapter 7. SEER takes a set of points from two images, and forms a large point cloud of potential 3D surface points by using a feature correspondence condition called the epipolar constraint. The surface is able to be extracted from the point cloud, because small neighbourhoods of surface points have different statistical characteristics to neighbourhoods of non-surface points resulting from incorrect correspondences.

³In a Matlab implementation, on a Pentium M 1.6GHz laptop computer

SEER is used to reconstruct a surface from the centroids of the tracked feature trajectories. This surface-construction procedure also identifies correspondences between the features between the two images, and hence between the two point trajectories in the two images. The motion of each 3D point is computed frame by frame by triangulating the corresponding points in each of the matched trajectories.

2.3 Thesis organisation

Because each chapter addresses problems from fairly distinct subfields of computer vision, relevant prior work will be discussed at the beginning of each chapter where appropriate, rather than in a separate thesis chapter. The separate components discussed in the previous sections are presented in Chapters 4 to 8. The full 3D reconstruction of motion of an actuated silicone gel breast phantom in a laboratory experiment is presented in Chapter 9, bringing together each stage of the imaging procedure, and showing how they relate.

Chapter 3

BACKGROUND

3.1 Notation

The natural mathematical setting for most of this research is the field of linear algebra. Geometric entities are represented as points, and most transformations are represented by matrices. Vectors in boldface type are always assumed to be column vectors $\mathbf{x} = (x, y)^T$. Sometimes it is convenient to refer to a geometrical entity independently of a particular coordinate frame. Typically, points and lines specified in this way will be referred to in italics, p, q, l . The matrix representing a transformation is written in typewriter font, H . The point, and its coordinates, in a minor abuse of notation, can be referred to interchangeably. For example the coordinates \mathbf{x} for a point will often be referred to as the point itself. The notation conventions follow the style used in the standard text for the field, Hartley and Zisserman [16]. Much of the material in this chapter is an abbreviated presentation of standard material from this text to provide a foundation for the reader.

3.2 Projective geometry of the plane

A full presentation of this material can be found in Hartley and Zisserman [16, §2.2]

3.2.1 Homogeneous representation of points and lines

The natural, familiar setting for a plane is in Euclidean space with Cartesian coordinates in \mathbb{R}^2 . A point is represented by a column vector $(x, y)^T$ and a line is represented by an equation $ax + by + c = 0$. The choice of a, b, c for a given line is not unique, the equation $(ka)x + (kb)y + (k)c = 0$, $k \neq 0$ represents the same line. The line can therefore be represented by a 3-vector $\mathbf{l} = (a, b, c)^T$, with any scalar multiple of this vector being regarded as equivalent.

The equivalence class of vectors under this equivalence relationship is called a *homogeneous* vector. Any particular vector $(a, b, c)^\top$ is a representative of the equivalence class. The set of equivalence classes of vectors $\mathbb{R}^3 \setminus (0, 0, 0)^\top$ forms the *projective space* \mathbb{P}^2 . The zero vector $(0, 0, 0)^\top$ is excluded as it does not represent any line.

A point (x, y) lies on the line $\mathbf{l} = (a, b, c)^\top$ if and only if $ax + by + c = 0$. This definition may be written as an inner product of vectors as $(x, y, 1)(a, b, c)^\top = \mathbf{x}^\top \mathbf{l} = 0$, with the point $(x, y)^\top$ being represented as a 3-vector $(x, y, 1)^\top$ by appending a 1 in the third coordinate. Again, the relationship is unchanged under scalar multiplication, $(kx, ky, k)\mathbf{l} = 0$, therefore points can also be represented by homogeneous vectors in \mathbb{P}^2 .

A distinction is made between the *homogeneous coordinates* $\mathbf{x} = (x_1, x_2, x_3)^\top$ for a point and its *inhomogeneous coordinates* $(x, y)^\top$. The inhomogeneous coordinates can be obtained by $(x, y)^\top = (x_1/x_3, x_2/x_3)^\top$, if $x_3 \neq 0$. If it is clear from context, the same notation \mathbf{x} will be used for both cases. If it is not clear, specific notation will be introduced where required.

A point can have homogeneous coordinates $(x_1, x_2, x_3)^\top$ where $x_3 = 0$, as long as at least one of x_1 and x_2 are nonzero. This point, when converted back to inhomogeneous coordinates in \mathbb{R}^2 is infinite. Therefore, \mathbb{P}^2 can be considered to be the plane, extended by a set of points at infinity, often called *ideal* points. It should be noted that in a pure study of projective geometry, the notion of ideal points doesn't exist. It arises in this context as a consequence of the mapping defined from \mathbb{P}^2 and \mathbb{R}^2 .

3.2.2 Intersection of points and lines

A collection of simple results about point and line intersections is presented in this section without proof. They provide basic building blocks for later use.

A point \mathbf{x} lies on a line \mathbf{l} if and only if:

$$\mathbf{l}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{l} = 0. \quad (3.1)$$

The line \mathbf{l} joining two points \mathbf{x}_1 and \mathbf{x}_2 is given by:

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2, \quad (3.2)$$

where \times denotes the standard vector cross product. Analogously, the intersection \mathbf{x} of two lines \mathbf{l}_1 and \mathbf{l}_2 is given by:

$$\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2. \quad (3.3)$$

Unlike in Euclidean geometry, every pair of distinct lines intersect in a point. There is no notion of parallel lines. Lines that are parallel in \mathbb{R}^2 intersect at an ideal point (with $x_3 = 0$).

Points and lines of \mathbb{P}^2 are duals. A theorem about points and lines can be transformed into a theorem about lines and points by simply changing all references to points into references to lines and vice versa.

3.2.3 A model for the projective plane

The set of points of the projective plane \mathbb{P}^2 can be visualised as the set of lines in \mathbb{R}^3 passing through the origin. Each of these lines is the set of points $k(x_1, x_2, x_3)^\top$ as k varies over $\mathbb{R} \setminus \{0\}$, for some x_1, x_2, x_3 . A line in \mathbb{P}^2 is similarly given by a plane in \mathbb{R}^3 passing through the origin. The relationship between the line in \mathbb{P}^2 and the plane in \mathbb{R}^3 can be seen by fixing a plane, typically $x_3 = 1$. The intersection of one of the lines of \mathbb{P}^2 with this plane gives a point $\mathbf{x} = (x, y, 1)^\top$. The intersection of a plane through the origin with this fixed plane gives a line \mathbf{l} . Coordinates in \mathbb{R}^2 for points of \mathbb{P}^2 are obtained by intersecting with the plane $x_3 = 1$ and then dropping the third coordinate.

3.2.4 Conics in the projective plane

A conic Q given by $ax^2 + by^2 + cxy + dx + ey + f = 0$ is also invariant to scalar multiplication. However, rather than giving it a homogeneous 6-vector, the relationship can be written as a symmetric matrix $\mathbf{Q} \in \mathbb{R}^{3 \times 3}$ in homogeneous coordinates given by:

$$\mathbf{Q} = \begin{bmatrix} a & c/2 & d/2 \\ c/2 & b & e/2 \\ d/2 & e/2 & f \end{bmatrix}, \quad (3.4)$$

with the equation for the conic becoming:

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = 0. \quad (3.5)$$

3.2.5 Projective transformations

A projective transformation is an invertible transformation that maps points in \mathbb{P}^2 to points in \mathbb{P}^2 and that maps lines to lines. Synonymous terms are *projectivity*, *collineation*, *homography*. In \mathbb{R}^2 a projective transformation $h: (x, y) \mapsto (x', y')$ is specified, using suggestive notation, by:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \quad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}. \quad (3.6)$$

Note that (3.6) is a homogeneous relationship. If all the h_{ij} are multiplied by a scalar, k , the resulting transformation is the same as the k normalises out between the numerator and denominator. In homogeneous coordinates, the expression is simpler. The transformation $h: (x_1, x_2, x_3)^\top \mapsto (x'_1, x'_2, x'_3)^\top$ can be represented by an invertible matrix

$\mathbf{H} \in \mathbb{R}^{3 \times 3}$ given by:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (3.7)$$

and the resulting point transformation is given by:

$$\mathbf{x}' = \mathbf{H}\mathbf{x}, \quad (3.8)$$

where $\mathbf{x} = (x_1, x_2, x_3)^\top$ and $\mathbf{x}' = (x_1, x_2', x_3')^\top$.

3.2.5.1 Computing a homography from four point correspondences

As the matrix is defined only up to scale, there are 8 degrees of freedom in specifying it, rather than 9. A point correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$ of points related by an unknown homography \mathbf{H} generates two linear constraints on the entries of \mathbf{H} , as can be seen by (3.6). Therefore, four point correspondences are sufficient to determine \mathbf{H} . If more correspondences are available, \mathbf{H} is determined by least squares using a procedure similar to that described later in §3.5.1

3.2.5.2 Projective transformation of lines

Under a point homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$, a line $\mathbf{l}^\top \mathbf{x} = 0$, in the new coordinates is given by $\mathbf{l}'^\top \mathbf{x}' = 0$. The line in the new coordinates is therefore given by:

$$\mathbf{l}' = \mathbf{H}^{-\top} \mathbf{l}. \quad (3.9)$$

3.2.5.3 Projective transformation of conics

Under a point homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$, a conic $\mathbf{x}^\top \mathbf{Q} \mathbf{x} = 0$ is given in the new coordinates by $\mathbf{x}'^\top \mathbf{H}^{-\top} \mathbf{Q} \mathbf{H}^{-1} \mathbf{x}' = 0$. The conic matrix therefore transforms as:

$$\mathbf{Q}' = \mathbf{H}^{-\top} \mathbf{Q} \mathbf{H}^{-1}. \quad (3.10)$$

3.2.5.4 The cross ratio

A configuration of concurrent lines on a plane, comprising a set of lines in 2D that meet at a single point, has a one-dimensional projective geometry \mathbb{P}^1 . An ordered set of four such lines have a *cross ratio*, which is an invariant under projective transformations of the plane. This cross ratio can also be regarded as the cross ratio of the four points of intersection of the set of lines with a distinct line that intersects the set of lines. The

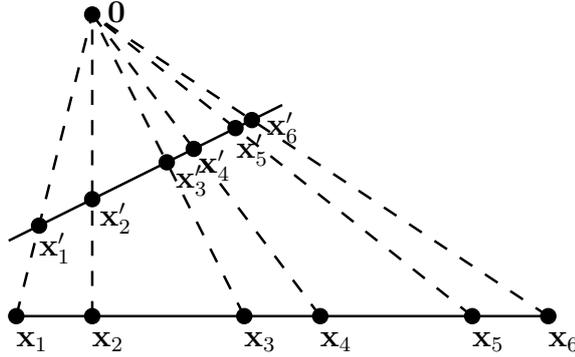


Figure 3.1: The projective geometry of concurrent lines

cross ratio has the same value no matter which intersection line is chosen. The cross ratio of four such points with coordinates in \mathbb{R}^2 is defined [16, p45]:

$$\text{Cross}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \frac{|\mathbf{x}_1\mathbf{x}_2||\mathbf{x}_3\mathbf{x}_4|}{|\mathbf{x}_1\mathbf{x}_3||\mathbf{x}_2\mathbf{x}_4|} \tag{3.11}$$

where

$$|\mathbf{x}_i\mathbf{x}_j| = \det \begin{bmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{bmatrix}. \tag{3.12}$$

Note that if the coordinates $x_{i2} = 1, x_{j2} = 1$ then $|\mathbf{x}_i\mathbf{x}_j|$ is the signed distance between the two points. Note also that the cross ratio is not invariant to the order of the points. Consider, for example, Fig. 3.1. The cross ratio of any ordered choice of four points, such as $\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5$, will be conserved under a projective transformation. The following cross ratios are therefore equal in this example:

$$\text{Cross}(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) = \text{Cross}(\mathbf{x}'_2, \mathbf{x}'_3, \mathbf{x}'_4, \mathbf{x}'_5) \tag{3.13}$$

3.3 Projective Space

In the same way as \mathbb{R}^2 was embedded in \mathbb{P}^2 , \mathbb{R}^3 can be embedded in \mathbb{P}^3 by using homogeneous coordinates. A point $(x, y, z)^\top \in \mathbb{R}^3$ can be written as a homogeneous vector $(x, y, z, 1)$. Projective space \mathbb{P}^3 can be conceptualised as the space of lines through the origin in \mathbb{R}^4 but this does not yield the same geometric intuition provided by the projective plane. The dual role of points and lines in \mathbb{P}^2 is replaced by the dual role of points and planes in \mathbb{P}^3 . A plane, given by $ax + by + cz + d = 0$ can be written as a homogeneous vector $\pi = (a, b, c, d)^\top$, and the condition for a point lying on a plane is $\pi^\top \mathbf{x} = \mathbf{x}^\top \pi = 0$. Lines in \mathbb{P}^3 do not have the simple formulation of lines in \mathbb{P}^2 . A line is represented by the join of two points, and there are a number of representations

of the resulting 6-parameter configuration, such as Plücker coordinates [30, A.II]. Any standard reference on projective geometry will have more information.

3.4 Single view geometry

3.4.1 Camera Model

Digital CCD cameras can be modelled accurately by projective pinhole cameras. The projection operation is illustrated in Fig. 3.2. In the coordinate system pictured, a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ is mapped to the point $(fX/Z, fY/Z, f)^\top$ on the retinal plane of the camera, and the image coordinates $\mathbf{x} = (fX/Z, fY/Z)^\top \in \mathbb{R}^2$ are obtained by dropping the final coordinate. If the points are written in homogeneous coordinates, by appending a ‘1’ as an additional coordinate, the projection operation can be written in terms of matrix multiplication as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \quad (3.14)$$

Note that in homogeneous coordinates, two points are considered equivalent if one is a scalar multiple of the other. Therefore, the point $(fX, fY, Z)^\top$ is equivalent to $(fX/Z, fY/Z, 1)^\top$. Equation (3.14) can be rewritten as:

$$\lambda \mathbf{x} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{X}, \quad (3.15)$$

where $\mathbf{x} = (x, y, 1)^\top$ are homogeneous image coordinates with λ accounting for the scale factor. The 3×3 matrix \mathbf{K} is given by $\mathbf{K} = \text{diag}(f, f, 1)$.

The image coordinates in (3.14) assume that the origin is the principal point, and Euclidean coordinates for x and y . In reality, the origin is typically considered to be the top left of the image, and the units are measured in pixels. For completeness, the model also allows that pixels be skew, and thus not rectangular. This behaviour is very unlikely to happen in many cases, as CCD sensors are manufactured to high precision, so if the skew parameter is not constrained to be zero, it is typically very small when the calibration is performed. Information about the origin, pixel scales, and skew can be embedded in the matrix \mathbf{K} as follows:

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.16)$$

where α_x, α_y are the scale factors representing pixel size, s is the skew parameter

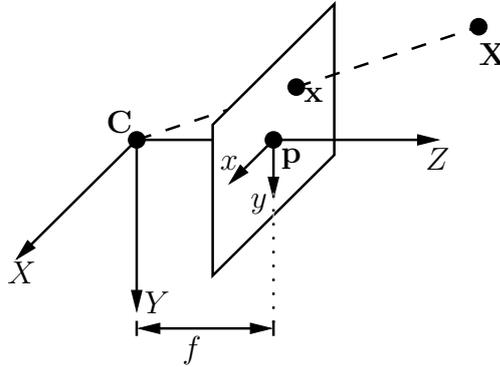


Figure 3.2: Pinhole camera geometry. \mathbf{C} is the camera centre, \mathbf{p} is the principal point. The 3D coordinate system is placed with the origin at \mathbf{C} and the Z axis aligned with the optical axis of the camera. The retinal plane is the plane $Z = f$ where f is the focal length of the camera. A point \mathbf{X} projects to a point \mathbf{x} in the image by intersecting the line joining \mathbf{C} and \mathbf{X} with the plane

(almost always near zero), and $(x_0, y_0)^\top$ are the image coordinates of the principal point. The parameters of \mathbf{K} are the *intrinsic* parameters of the camera model. It is often more convenient to use an arbitrary three-dimensional world coordinate frame that is not the frame depicted in Fig. 3.2. Assume that the transformation from world frame to the 3D camera frame is given by:

$$\mathbf{X}_c = \mathbf{R}\mathbf{X}_w + \mathbf{t}. \quad (3.17)$$

where coordinates are in \mathbb{R}^3 , \mathbf{R} is a 3×3 rotation matrix, and \mathbf{t} is the location of the world origin in the camera frame of reference. The parameters \mathbf{R} and \mathbf{t} , representing the position and orientation of the camera in space, are the *extrinsic* parameters of the camera. In homogeneous coordinates the coordinate transformation can be written as a matrix multiplication:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \quad (3.18)$$

The corresponding mapping to (3.14), incorporating the updated \mathbf{K} can therefore be written in homogeneous coordinates:

$$\lambda \mathbf{x} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_w = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \mathbf{X}_w. \quad (3.19)$$

The matrix $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]$ is the projection matrix for the camera. The w subscript from \mathbf{X}_w will be dropped from here, as 3D coordinates will always be understood to be in the world frame unless specified. Therefore, the projection operation for a camera given by

intrinsic parameter matrix K and extrinsic parameters R, \mathbf{t} in homogeneous coordinates is defined:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K[R \mid \mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (3.20)$$

or, more simply:

$$\lambda \mathbf{x} = P\mathbf{X}. \quad (3.21)$$

3.5 Calibration

A camera is considered to be calibrated when the parameter matrices K, R, \mathbf{t} and hence P are known. A standard method of camera calibration, called resection, is computing the projection matrix P by least squares from a number of world-image correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$. Constraints on P can be built up from (3.21) for each correspondence. Once P is known, K, R , and \mathbf{t} can be computed by matrix decomposition techniques from linear algebra.

3.5.1 Camera resection from point correspondences

The process of estimating the camera projection matrix P from corresponding 3-space and image points is known as camera *resectioning*. The algorithm presented in this section is a summary of the development in [16]. The problem is

Problem 3.1. *Given n correspondences $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$ between 3-space points \mathbf{X}_i and image points \mathbf{x}_i , estimate the 3×4 matrix P that best satisfies the homogeneous relation $P\mathbf{X}_i = \lambda\mathbf{x}_i$ for all i .*

Given a point correspondence $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$, the quantities $P\mathbf{X}_i$ and \mathbf{x}_i are related by multiplication by scalar multiplication by an unknown scalar λ , and are therefore parallel. The scalar λ can be eliminated by taking the cross product $\mathbf{x}_i \times P\mathbf{X}_i = \mathbf{0}$. Writing $P\mathbf{X}_i$ as:

$$P\mathbf{X}_i = \begin{pmatrix} \mathbf{p}^{1\top} \mathbf{X}_i \\ \mathbf{p}^{2\top} \mathbf{X}_i \\ \mathbf{p}^{3\top} \mathbf{X}_i \end{pmatrix}, \quad (3.22)$$

where $\mathbf{p}^{j\top}$ is the j -th row of the matrix P and writing \mathbf{x}_i as $(x_i, y_i, w_i)^\top$, the cross product can be written explicitly as:

$$\mathbf{x}_i \times P\mathbf{X}_i = \begin{pmatrix} y_i \mathbf{p}^{3\top} - w_i \mathbf{p}^{2\top} \\ w_i \mathbf{p}^{1\top} - x_i \mathbf{p}^{3\top} \\ x_i \mathbf{p}^{2\top} - y_i \mathbf{p}^{1\top} \end{pmatrix}. \quad (3.23)$$

Since $\mathbf{p}^{j\top}\mathbf{X}_i = \mathbf{X}_i^\top\mathbf{p}^j$ for $j = 1, \dots, 3$, this gives a set of three equations in the entries of \mathbf{P} which can be rewritten:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \\ -y_i\mathbf{X}_i^\top & x_i\mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0}. \quad (3.24)$$

Note that the third row of the matrix is linearly dependent on the first two, as it can be written as the sum of $-y_i/w_i$ times the second row and $-x_i/w_i$ times the first. Omitting, then, the third equation, the set of equations becomes:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0}. \quad (3.25)$$

which is written in compact notation as:

$$\mathbf{A}_i\mathbf{p} = \mathbf{0} \quad (3.26)$$

where \mathbf{A}_i is the 2×12 matrix of (3.25) and $\mathbf{p} = (\mathbf{p}^{1\top}, \mathbf{p}^{2\top}, \mathbf{p}^{3\top})^\top$ consists of the rows of \mathbf{P} stacked into a column vector, so that:

$$\mathbf{p} = (p_{11}, p_{12}, \dots, p_{33}, p_{34})^\top \quad \text{where} \quad \mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}. \quad (3.27)$$

Since \mathbf{P} needs only to be determined up to scale, there are 11 unknowns, requiring 11 independent constraints on its entries to obtain a unique solution. Each constraint $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$ contributes two independent linear constraints on the entries of \mathbf{P} so at least 6 correspondences are needed to determine \mathbf{P} .

For each correspondence $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$, $i = 1, \dots, n$, the matrix \mathbf{A}_i of (3.25) is formed and these matrices are stacked into the $2n \times 12$ matrix \mathbf{A} defined:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_n \end{bmatrix}. \quad (3.28)$$

The correspondence relationship can thus be written:

$$\mathbf{A}\mathbf{p} = \mathbf{0} \quad (3.29)$$

If the correspondences are not exact, where for example the measured points $\mathbf{x}_i, \mathbf{X}_i$

are corrupted by noise, and there are $n \geq 6$ point correspondences, there will not be an exact solution to $\mathbf{A}\mathbf{p} = \mathbf{0}$. An estimate of \mathbf{P} is therefore obtained by minimising an algebraic or geometric error.

3.5.1.1 Algebraic error minimisation

If \mathbf{p} is constrained to have unit Euclidean norm, $\|\mathbf{p}\| = 1$, the vector \mathbf{p} , and hence \mathbf{P} , can be estimated by solving the constrained least squares problem:

$$\min_{\mathbf{p}} \|\mathbf{A}\mathbf{p}\|^2, \quad \text{subject to} \quad \|\mathbf{p}\| = 1. \quad (3.30)$$

The residual $\mathbf{A}\mathbf{p}$ is called the *algebraic error*. The solution to (3.30) is found by computing the singular value decomposition (SVD) of \mathbf{A} , $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and choosing the last column of \mathbf{V} to be the estimate of \mathbf{p} . \mathbf{P} is then formed by unstacking \mathbf{p} into \mathbf{P} as in (3.27). Note that the Euclidean norm constraint is not the only possible constraint on \mathbf{p} . Other possible constraints include $\|\hat{\mathbf{p}}^3\| = 1$, where $\hat{\mathbf{p}}^3$ is the vector $(p_{31}, p_{32}, p_{33})^T$ (see [16, p183]).

3.5.1.2 Geometric error minimisation

The disadvantage of minimising the Algebraic error (3.30) is that it is not clear what exactly is being minimised geometrically or in what sense it is a best estimate. If the calibration object is manufactured to high precision the errors in the world locations \mathbf{X}_i can be assumed to be negligible. If it is then assumed that image point measurement errors obey a Gaussian error model, or more specifically a zero-mean isotropic Gaussian distribution with uniform variance σ^2 , then if the true point is $\bar{\mathbf{x}} = \mathbf{P}\bar{\mathbf{X}}$ the probability density function for a measured point \mathbf{x} is:

$$p(\mathbf{x}) = \left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} e^{-d(\mathbf{x}, \bar{\mathbf{x}})^2 / (2\sigma^2)}, \quad (3.31)$$

where $d(\cdot, \cdot)$ is the Euclidean distance between two image points. The probability density function for obtaining the set of correspondences $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$ for a given projection matrix \mathbf{P} , assuming that the noise on each measurement is independent and zero noise in the world coordinates, is thus given by:

$$p(\{\mathbf{x}_i\}|\mathbf{P}) = \prod_i \left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} e^{-d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2 / (2\sigma^2)}. \quad (3.32)$$

This expression is called the *likelihood* of the set of correspondences. The *log-likelihood* of the set of correspondences is defined:

$$\log p(\{\mathbf{x}_i\}|\mathbf{P}) = -\frac{1}{2\sigma^2} \sum_i d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2 + \text{constant}. \quad (3.33)$$

The *Maximum Likelihood estimate* (MLE) of the projection matrix $\hat{\mathbf{P}}$, maximises this log-likelihood, so it minimises the *geometric error*:

$$\sum_i d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2, \quad (3.34)$$

Equation (3.34) is simply the sum of squared distances from the measured image points to their true locations. The MLE of the projection matrix, $\hat{\mathbf{P}}$, is thus given by:

$$\hat{\mathbf{P}} = \arg \min_{\mathbf{P}} \sum_i d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2, \quad (3.35)$$

i.e. $\hat{\mathbf{P}}$ is the matrix \mathbf{P} that minimises the geometric error. This nonlinear least squares problem can be solved using a standard nonlinear least squares algorithm like the Levenberg-Marquardt or Gauss-Newton algorithms, employing the solution to the algebraic error problem (3.30) as an initial estimate.

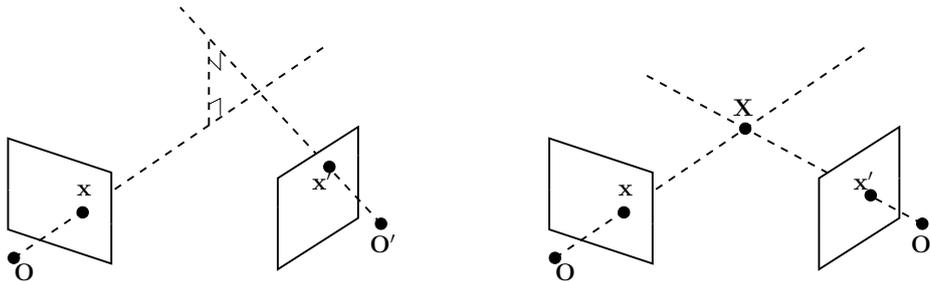
3.5.1.3 Normalisation

Typically it is the case that image points $\mathbf{x}_i = (x_i, y_i, w_i)^\top$ are of the form $(u, v, 1)^\top$ where u, v are pixel coordinates, e.g. $\mathbf{x}_i = (500, 400, 1)$. Using coordinates like these can result in a badly conditioned system in (3.25, 3.30) as the x, y coordinates are much larger in magnitude than the w coordinate. Because the geometric error (3.34) is invariant to similarity transformations of both image and world coordinates, the conditioning of the problem can be improved by transforming the coordinates to a more uniform space. This transformation is done by transforming the image coordinates by a similarity transformation, \mathbf{T} , such that they have mean 0 in each coordinate and mean distance to the origin of $\sqrt{2}$. The “average” point is then $(1, 1, 1)^\top$. The transformed image points $\tilde{\mathbf{x}}_i$ are given by:

$$\tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i. \quad (3.36)$$

The world coordinates are similarly transformed by a similarity transformation \mathbf{U} to have zero mean and mean distance to the origin of $\sqrt{3}$ so that the “average” point is $(1, 1, 1, 1)^\top$. These transformed world coordinates $\tilde{\mathbf{X}}_i$ are given by:

$$\tilde{\mathbf{X}}_i = \mathbf{U}\mathbf{X}_i. \quad (3.37)$$



(a) Points \mathbf{x} and \mathbf{x}' do not satisfy the epipolar constraint because the rays backprojected from their respective optical centres do not intersect

(b) Points \mathbf{x} and \mathbf{x}' satisfy the epipolar constraint because the rays backprojected from their respective optical centres intersect at a point \mathbf{X} in space

Figure 3.3: Epipolar constraint

The camera resection (3.35) can then be performed in these normalised coordinates. The resulting projection matrix \tilde{P} can then be modified to give the projection matrix P for the original coordinates as:

$$P = T^{-1}\tilde{P}U. \quad (3.38)$$

3.5.2 Factorising the projection matrix

Once the projection matrix P is known, the intrinsic parameter matrix K and the extrinsic parameters R and \mathbf{t} can be found. The projection matrix P can be written:

$$P = K[R \mid \mathbf{t}] = [KR \mid K\mathbf{t}]. \quad (3.39)$$

The submatrix KR can be factorised into K and R by the RQ factorisation [16, A4.1], a variant of the well-known QR factorisation. Once K is known, \mathbf{t} can then be found by multiplying $K\mathbf{t}$ by K^{-1} , as K is always invertible.

3.6 Multiple view geometry

3.6.1 Epipolar geometry

The *epipolar constraint* is a geometric constraint on points in two images that essentially determines whether it is *possible* for two points \mathbf{x}, \mathbf{x}' to be the images of a single world point. If rays are back-projected through \mathbf{x} and \mathbf{x}' from their respective optical centres, then if the rays meet in space at a point, \mathbf{x} and \mathbf{x}' are said to satisfy the epipolar constraint. This interpretation of the epipolar constraint is depicted schematically in Fig. 3.3.

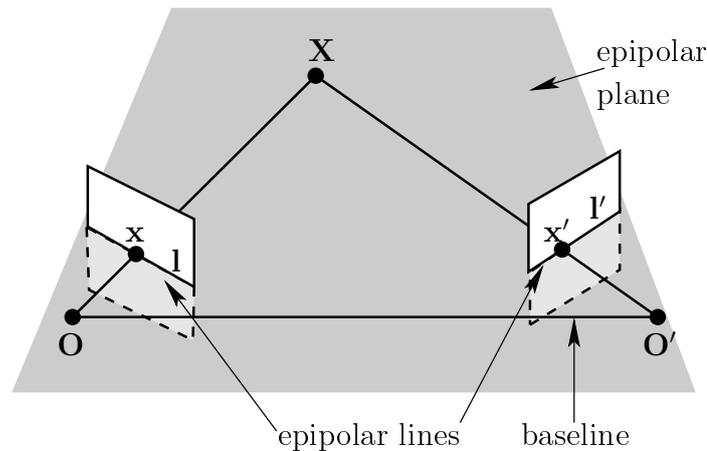


Figure 3.4: Epipolar geometry: The intersection of the epipolar plane constructed from the baseline and \mathbf{X} with the two image planes gives two epipolar lines \mathbf{l}, \mathbf{l}' . Potential matches in the second image to \mathbf{x} must fall on \mathbf{l}' , and vice versa.

Because the rays passing through \mathbf{x}, \mathbf{x}' meet at a single point, the optical centres \mathbf{O}, \mathbf{O}' , the image points \mathbf{x}, \mathbf{x}' and the world point \mathbf{X} are all coplanar. The plane containing these points is known as an *epipolar plane* as illustrated in Fig. 3.4. The intersection of this plane (shown in grey) with the two image planes gives two lines \mathbf{l}, \mathbf{l}' , called *epipolar lines*. All epipolar planes contain the *baseline*, which is the line joining \mathbf{O} and \mathbf{O}' , and therefore there is a one-parameter family of epipolar planes for a given stereo camera configuration.

Any two points on \mathbf{l} and \mathbf{l}' satisfy the epipolar constraint, as the rays back-projected through them are coplanar and therefore intersect (at ‘infinity’ if the rays are parallel). Given a point \mathbf{x} in the first image, the corresponding epipolar line \mathbf{l}' can be found (conceptually) by constructing the epipolar plane from the points $\mathbf{O}, \mathbf{O}', \mathbf{x}$, and intersecting this plane with the second image plane. The corresponding point \mathbf{x}' must then lie on \mathbf{l}' . The epipolar geometry is depicted in Fig. 3.4.

3.6.2 The Fundamental matrix

Given a feature \mathbf{x} in one image obtained from a camera with projection matrix \mathbf{P} , the corresponding epipolar line \mathbf{l}' in a second camera with projection matrix \mathbf{P}' is the image of the ray back-projected from the \mathbf{O} through \mathbf{x} in the second image.

The ray back-projected from \mathbf{x} is found by finding the set of points which project to \mathbf{x} , which entails solutions for \mathbf{X} of the equation:

$$\mathbf{P}\mathbf{X} = \mathbf{x} \quad (3.40)$$

Note that homogeneous coordinates are being used exclusively in this section. Two points on this line are the camera centre \mathbf{O} , and $\mathbf{P}^+\mathbf{x}$, where $\mathbf{P}^+ = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}$ is the

pseudo-inverse of P , for which $PP^+ = I$, as P is full rank [14, §5.5.4]. The point $P^+\mathbf{x}$ is on the ray because it projects to \mathbf{x} , i.e. $PP^+\mathbf{x} = \mathbf{x}$. The images of the points \mathbf{O} and $P^+\mathbf{x}$ from the second camera are the points $P'\mathbf{O}$ and $P'P^+\mathbf{x}$. The epipolar line is the join of these points, defined using (3.2) as:

$$\mathbf{l}' = (P'\mathbf{O}) \times (P'P^+\mathbf{x}). \quad (3.41)$$

The camera centre \mathbf{O} can be found by solving $P\mathbf{O} = \mathbf{0}$. The point $\mathbf{e}' = P'\mathbf{O}$ is called an *epipole*. Let $[\mathbf{e}']_{\times}$ be the 3×3 matrix representation of the cross product operation with \mathbf{e}' , i.e. $[\mathbf{e}']_{\times}\mathbf{y} = \mathbf{e}' \times \mathbf{y}$ for all \mathbf{y} . Then, (3.41) can be rewritten:

$$\mathbf{l}' = [\mathbf{e}']_{\times}P'P^+\mathbf{x} = \mathbf{F}\mathbf{x} \quad (3.42)$$

where $\mathbf{F} = [\mathbf{e}']_{\times}P'P^+$ is a 3×3 matrix called the *fundamental matrix* of the stereo configuration.

A point \mathbf{x}' satisfies the epipolar constraint with \mathbf{x} , if and only if it lies on the epipolar line \mathbf{l}' , represented algebraically by:

$$\mathbf{x}'^T \mathbf{l}' = 0. \quad (3.43)$$

The epipolar constraint is then written in terms of the fundamental matrix as:

$$\mathbf{x}'^T \mathbf{F}\mathbf{x} = 0 \quad (3.44)$$

3.6.3 Triangulation

The process of computing the 3D position of a point from its image location in two images, from cameras with projection matrices P, P' , is called *triangulation*. Considering Fig. 3.3b, a point \mathbf{X} that projects both to \mathbf{x} and \mathbf{x}' would imply:

$$\begin{aligned} P\mathbf{X} &= \lambda\mathbf{x} & \text{and} \\ P'\mathbf{X} &= \lambda'\mathbf{x}'. \end{aligned} \quad (3.45)$$

In the presence of noise, these relationships will not be exactly satisfied, and \mathbf{X} must be estimated.

The homogeneous scale factors λ, λ' are normalised out by taking the cross product $\mathbf{x} \times (P\mathbf{X}) = \mathbf{0}$. The cross product can be rewritten in full as:

$$\begin{aligned} x(\mathbf{p}^3{}^T \mathbf{X}) - (\mathbf{p}^1{}^T \mathbf{X}) &= 0 \\ y(\mathbf{p}^3{}^T \mathbf{X}) - (\mathbf{p}^2{}^T \mathbf{X}) &= 0 \\ x(\mathbf{p}^2{}^T \mathbf{X}) - y(\mathbf{p}^1{}^T \mathbf{X}) &= 0 \end{aligned} \quad (3.46)$$

where $\mathbf{x} = (x, y, 1)^\top$, and \mathbf{p}^i are the rows of \mathbf{P} . The third equation is linearly dependent on the first two, so the first two equations supply two linear constraints on the entries of \mathbf{X} . Similarly, $\mathbf{x}' \times \mathbf{P}'\mathbf{X} = \mathbf{0}$ supplies two further linear constraints. The constraints can be stacked in a matrix \mathbf{A} defined:

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} - \mathbf{p}^{2\top} \\ x'\mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \\ y'\mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \end{bmatrix} \quad (3.47)$$

and used to solve for \mathbf{X} in the equation:

$$\mathbf{A}\mathbf{X} = \mathbf{0}. \quad (3.48)$$

To obtain a unique solution for \mathbf{X} , which is only defined up to scale, \mathbf{X} is constrained to have unit (Euclidean) norm. The least squares estimate of the point \mathbf{X} in homogeneous coordinates is then found by computing the SVD, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and choosing \mathbf{X} to be the column of \mathbf{V} corresponding to the smallest singular value.

3.7 Summary

This chapter has provided the basic foundational principles and relationships that underpin the algorithms developed in the remainder of the thesis. The nomenclature introduced throughout this chapter will be used in all subsequent chapters, unless explicitly stated. In particular, it is important to reiterate that homogeneous and inhomogeneous coordinates for a point may be referred to by the same vector (e.g. \mathbf{x}) if the context is clear.

For further information, there are a number of excellent computer vision texts. Examples of which include [9, 16, 25, 38].

Chapter 4

CAMERA CALIBRATION: INTRODUCTION AND INITIAL APPROACH

4.1 Introduction

Accurate camera calibration is an essential component of most 3D computer vision systems. Calibration provides a mathematical model of the camera that describes how points in the real world are related to image (pixel) locations. Digital CCD (Charge-Coupled Device) cameras can be modelled by models of varying degrees of complexity. The operation of an ideal pinhole camera, described in §3.4.1, is linear in homogeneous coordinates, and is represented by simple matrix multiplication. Modern digital cameras fit the pinhole camera model surprisingly well, as is shown later in this research. More sophisticated models include parameters to describe the radial and tangential distortion induced by the lens, which is particularly evident in images from cameras with wide-angle lenses.

The parameters of a camera model can be separated into two groups, extrinsic and intrinsic parameters. The extrinsic parameters of a camera describe its position and orientation with respect to a fixed world frame of reference. The intrinsic parameters of the camera describe how coordinates on the retinal plane of the camera (§3.4.1) are mapped to pixel coordinates. Typical intrinsic parameters include the principal point, pixel sizes, pixel skew, and focal length.

Just as there is a large variety in the complexity of camera models available, the calibration requirements for computer vision applications vary appreciably. Dependent parameters include the camera configuration, whether the cameras are static or moving, whether the object being imaged is at close range, the desired reconstruction accuracy, whether a full 3D reconstruction is necessary, and many more. Hence, different applications require substantially different calibration approaches, as is evidenced by the

large body of literature on the subject, e.g. [39, 41, 42].

4.1.1 Close-range camera calibration with a rig

The standard technique for calibrating cameras at close range in a controlled environment is by use of a calibration rig. The calibration rig is constructed to a high level of accuracy. Images are taken of the calibration rig, and by accurately measuring the image locations of the marker points on the rig and matching these with their known world coordinates, the camera model can be computed very accurately.

There are two main approaches to the calibration rig problem, both of which are very accurate. The first, more conventional, method involves using a three-dimensional rig with accurately placed markers on it. The camera calibration is computed from feature correspondences obtained from a single image of the rig. The second approach is attractive because it does not require a specialised calibration object. The rig is a planar chequerboard pattern, known as a Tsai grid [39]. Rather than computing the calibration from a single image, which is not possible, images are taken of the chequerboard held at a number of different orientations. The calibration is computed by combining measurements from each image.

The Tsai grid method becomes logistically more difficult to implement with an array of multiple cameras, as numerous images need to be taken and combined from each camera. Further, in most software implementations, a significant amount of user input is required. For a system that needs to be calibrated frequently, as the DIET system may well need to be, this method becomes cumbersome. Therefore, the chosen method was the three-dimensional rig approach.

There are three main subproblems associated with this calibration procedure:

1. Accurately measuring the image locations of the feature points
2. Finding correspondences between image and feature points
3. Computing the camera model

The first two are dependent on the design of the calibration object, the third is dependent on the choice of camera model.

4.1.2 DIET requirements

The DIET system requires sufficient cameras to image the entire breast surface. Experiments performed, discussed later in the thesis in Chapter 9, suggest that the minimum number of cameras required is five, with a greater number being desirable. Each of these cameras needs to be fully calibrated. Since one of the engineering requirements

Algorithm 4.1 High-level camera calibration procedure

1. Place the calibration object in the centre of the camera viewing field
 2. Adjust lighting if necessary
 3. Take an image from each camera of the object
 4. Lock the camera lens settings: zoom, focus, aperture
 5. Compute the camera calibration of each camera
 6. Identify any problems with the camera calibration, and repeat the procedure if necessary
 7. Remove the calibration object
-

of the DIET system is that the system be portable and easy to operate, it is important that the procedure to calibrate the cameras is as simple and fast as possible. A calibration procedure taking half an hour to calibrate all the cameras would make the system much less effective for portable screening on the road, and is therefore unacceptable. An idealised calibration procedure is shown in Algorithm 4.1. Ideally, the entire procedure would take no more than a few minutes, allowing rapid setup of the DIET system. With the eventual DIET system in mind, the following desiderata for the camera calibration algorithm were decided upon, in approximate order of importance:

1. **Accuracy:** The camera calibration should sufficiently accurate that the motion measurement error is minimal
2. **Viewpoint:** It should be possible to calibrate cameras from a wide range of viewpoints with respect to the calibration object, allowing the calibration of a large array of cameras without moving the calibration object
3. **Robustness:** The algorithm should be robust to natural variations in lighting conditions
4. **Speed:** Calibration of a single camera should be as fast as possible
5. **Automation:** Calibration should be fully automatic, requiring no user input

4.1.3 Calibration rig design

Three calibration cubes were developed. The first, depicted in Fig. 4.1, was constructed in preliminary experiments for the DIET system [4]. The calibration algorithm was rewritten in its entirety and is presented in §4.2. The resulting algorithm satisfied Desideratum 1, but was very slow due to the computationally intensive image processing procedures used. Two other drawbacks were that three faces of the cube were

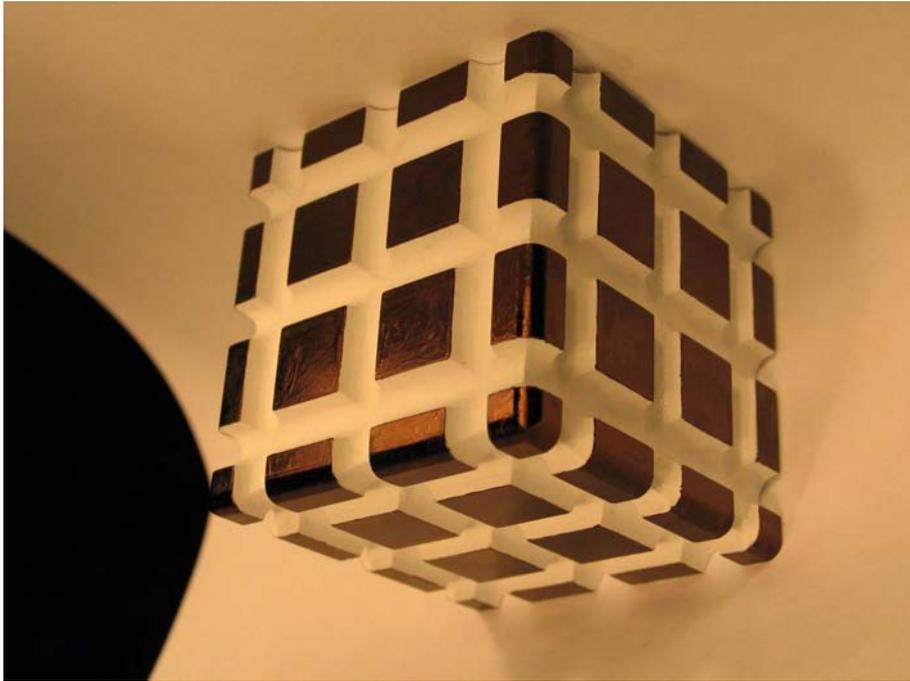


Figure 4.1: First calibration cube, on a white background

required to be clearly visible, limiting the possible camera viewing angles, and the algorithm was not fully automatic, requiring user input to determine which faces of the cube were visible. This method was replaced when the experimental system was expanded to more than two cameras, as successfully calibrating all of the cameras became prohibitively difficult. Finding positions for the cameras and the cube that gave satisfactory viewpoints from each camera was also a major cause of difficulty.

The second two cubes developed use the same calibration algorithm, and are depicted in Figs. 4.2 and 4.3. This new algorithm, described in Chapter 5, allows for fully automatic calibration from a much wider range of camera viewpoints. Using circles as point features made the measurement of image features more accurate, and easier.

The first of the two ‘Die’ cubes shown in Fig. 4.2 had some problems with segmenting the cube from the background, due to shadowing from the lights, and so failed to satisfy Desiderata 3. There were also fewer point features than is desirable for accurate calibration, particularly when only two faces of the cube were visible.

The final cube developed, shown in Fig. 4.3, added extra point features to each face for robustness in the camera matrix estimation. It was chosen to be white so that a black background could be used, virtually eliminating the problems of shadowing. The edge length of the cube was increased from 54mm to 100mm so that camera calibration was computed from measurements taken from a larger proportion of the image in the experimental setup.

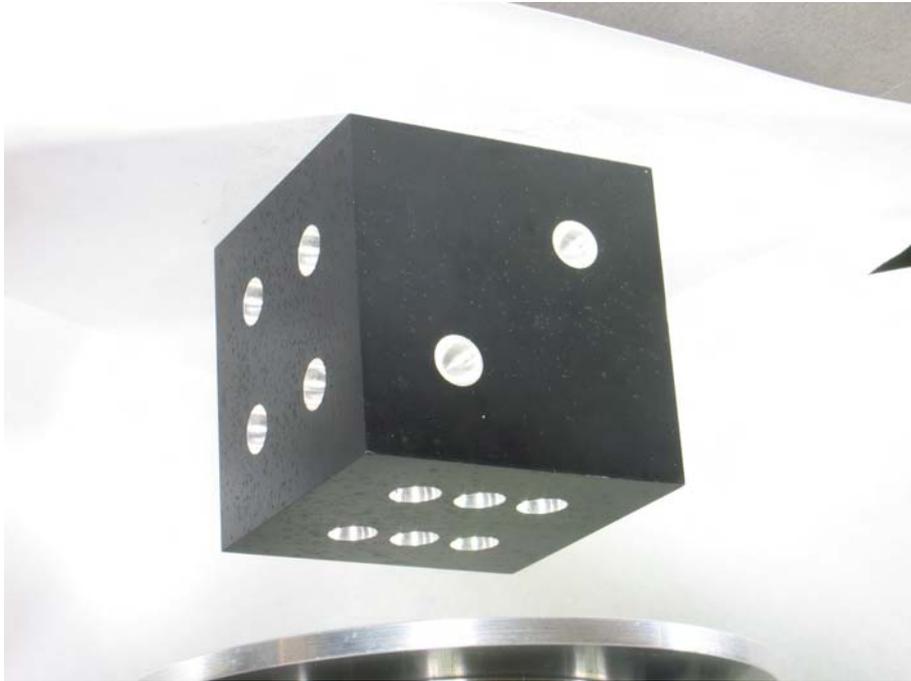


Figure 4.2: Second calibration cube

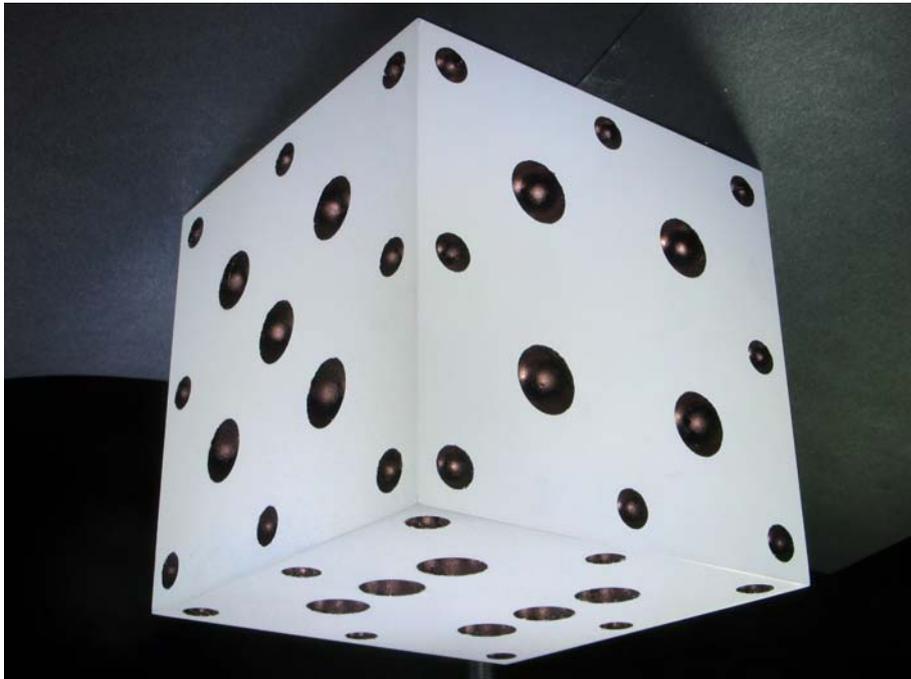


Figure 4.3: Third calibration cube

4.2 Calibration using the grid-based cube

The calibration algorithm used to calibrate cameras from images of a three-dimensional rig is camera resection, as described in §3.5.1. For the calibration cube depicted in Fig. 4.1, the feature points are defined to be the 36 interior corners on each of the three visible faces of the cube, giving 108 feature points from which to compute the model.

Before the model can be computed, there are two subproblems that need to be addressed:

1. **Identification:** Finding the 108 feature points, given an image
2. **Correspondence:** Finding correspondences between the measured points and the known locations of the faces of the cube

These two problems are addressed in the following sections.

4.2.1 Feature identification

The overall procedure for the feature identification problem is shown in Algorithm 4.2 and described in the subsequent subsections.

4.2.1.1 Line detection

The first step in finding line features in an image is to detect the image edge pixels. In this case, the edges are the boundary between the light and dark regions of the calibration object. This task can be done by converting the image to grayscale and then using a standard edge detection algorithm, such as the Canny [3] or Sobel [36] edge detectors. Due to the high contrast of the calibration object, the choice of edge detection algorithm is not particularly important. The Sobel edge detector implementation from Matlab's Image Processing Toolbox was used, which computes the threshold for the Sobel operator automatically. An example of an edge image produced by the edge detection step is shown in Fig. 4.4.

The next step in detecting line features is to examine the edge image of Fig. 4.4 and find the presence of image lines. The standard technique for this task is to use the Hough transform, or the related Radon transform. The space of all possible image lines can be parametrised by two parameters, for example a pair (ρ, θ) representing the line $x \cos \theta + y \sin \theta = \rho$. In the Hough transform, the two dimensional parameter space parametrising all possible image lines is discretised, and an accumulator array is constructed containing a bin for each discrete parameter pair. For each edge pixel a vote is given to each point in parameter space corresponding to a line which passes sufficiently close to the centre of the pixel. Points in the parameter space corresponding

Algorithm 4.2 Finding the 108 feature points in an image of the grid cube

1. Find image lines
 - (a) Perform edge detection on the grayscale image
 - (b) Take the Radon transform of the edge image
 - (c) Use nonmaxima suppression to find peaks in the Radon transform, which correspond to image lines
 2. Use MMRANSAC, a robust model estimation algorithm, to identify three vanishing points and to classify the inlying lines into three mutually perpendicular directions in space
 3. For each pair of vanishing points (defining one face of the cube):
 - (a) Construct the vanishing line joining the two vanishing points
 - (b) For the star of lines emanating from each vanishing point:
 - i. Intersect each line with the perpendicular bisector of the vanishing point
 - ii. Discard the 6 farthest points from the vanishing line.
 - iii. Compute the cross-ratio invariant for each of the $\binom{n}{6}$ combinations of 6 of the n remaining points
 - iv. Choose the combination which is closest to the theoretical invariant (by Euclidean distance) and less than some threshold. If there are too few points, or there is no combination satisfying this, terminate with failure
 4. The 36 vertices for the face are the 6×6 intersections between each pair of 6 lines chosen
-

to image lines receive a large number of votes, one for each pixel lying on the line. Therefore, image lines can be detected as peaks in the accumulator array.

The Hough transform can be interpreted as a discretisation of the Radon transform [7], and implementations of the Radon transform for discrete images perform a similar task. The Radon transform, in contrast to the Hough transform, is defined in terms of line integrals of the image along each possible image line. In the Radon transform implementation on the Matlab software platform, each pixel is split into four subpixels, and for lines of a given slope, the (quarter) vote for each subpixel is split proportionally between the two lines that pass nearest to it [26]. This implementation of the Radon transformation is therefore a much smoother transformation than the Hough transform, at the expense of considerably more computational time. This smoothness makes the peaks easier to localise, as shown in Fig. 4.5, and thus the Radon transform was chosen for the line detection stage. Note that in most applications the Hough transform performs perfectly adequately. The Matlab implementation of the Radon transform is simply better suited to the images used in this specific project.

The parametrisation of image lines used by the Radon transform is shown in

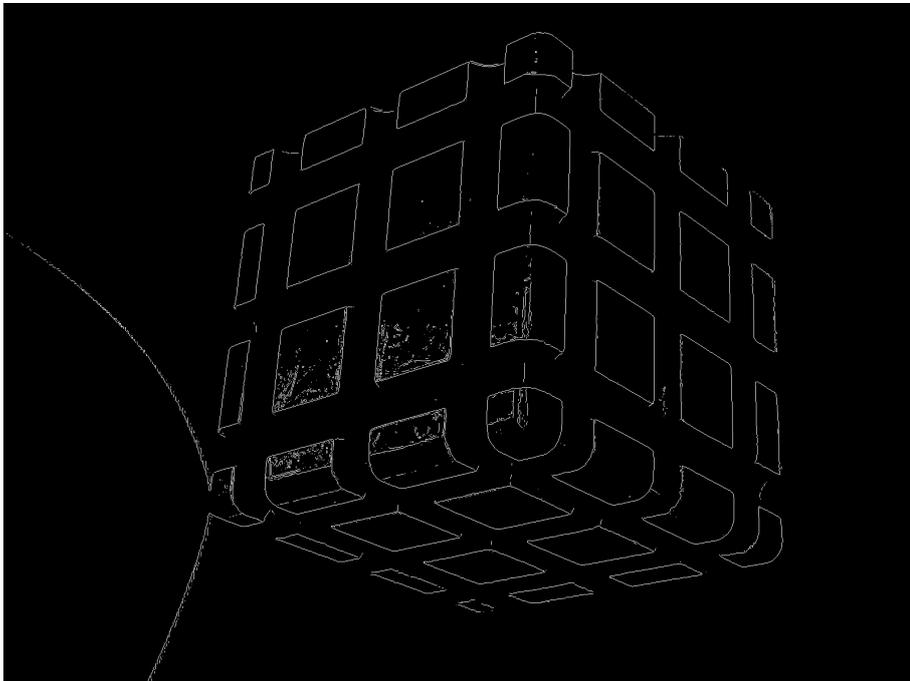
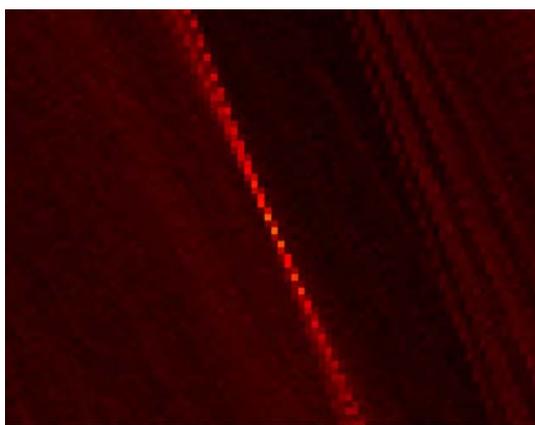
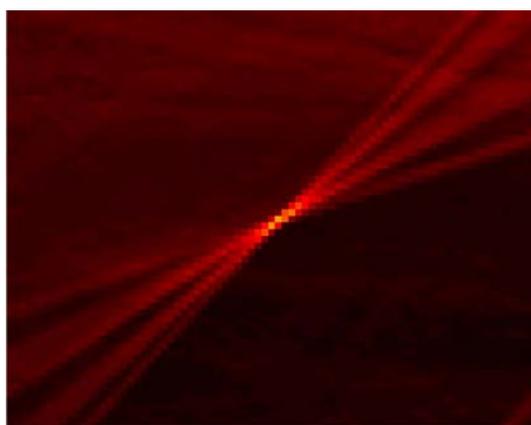


Figure 4.4: Edges of the first calibration cube of Fig. 4.1 found by the Sobel edge detector



(a) Hough transform peak



(b) Radon transform peak

Figure 4.5: Close up of a peak from the Hough and Radon transforms of the edge image in Fig. 4.4 for essentially the same parameter space. Note how the Radon image is smoother, and the peak more easily localised

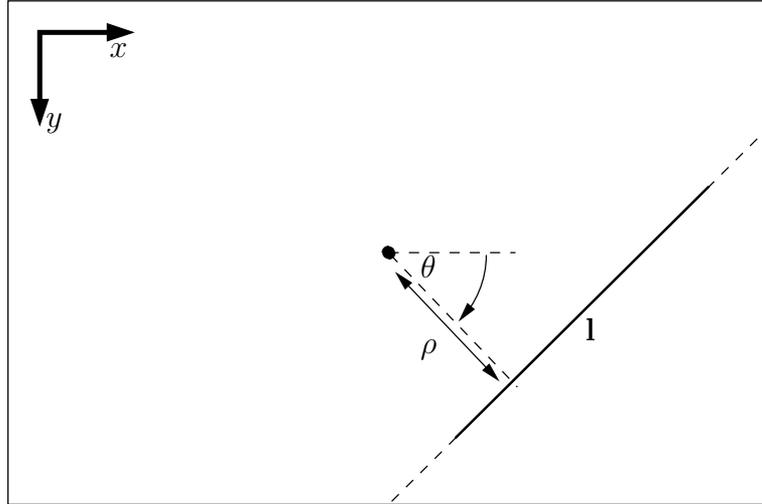


Figure 4.6: Line parametrisation used for the Radon transform. The image line **l** (shown in bold) is parametrised by (ρ, θ) where ρ is the perpendicular distance from **l** to the centre of the image, and θ the clockwise angle from the horizontal to the corresponding normal to **l**

Fig. 4.6. This parametrisation is discretised by choosing uniformly spaced θ values over a 180° range, and uniformly spaced ρ values to cover the image. For the 2 megapixel (1600×1200 pixels) images used in Fig. 4.1, the θ spacing was chosen to be 0.1° and the ρ spacing to be 1 pixel.

The peaks in the Radon transform space, which will often be referred to as the *Radon image* in this thesis, such as the one depicted in Fig. 4.5b are found by a technique called nonmaxima suppression. The peaks are found by performing a grayscale dilation on the Radon image with an approximately circular neighbourhood of radius r pixels. A grayscale dilation is where each pixel in the output image is set to the maximum value of the pixels in a specified neighbourhood of the corresponding input pixel in the input image. All points where the dilated image matches the original image are local peaks, and all peaks over a certain threshold are chosen. The size of neighbourhood determines the minimum distance two peaks may be from each other. A neighbourhood of radius r pixels means that no two peaks can be within r pixels of each other. The implementation used was due to Kovesi [20] and also allows for subpixel localisation of the peaks.

Let $I(\theta, \rho)$ be the value of the Radon image at coordinates θ, ρ , and let $\Delta\theta, \Delta\rho$ be the parameter increments at adjacent points. For a peak in the Radon image at θ, ρ :

1. A quadratic $Q_\theta(\theta)$ is fitted to the three points consisting of the image peak and the two horizontally adjacent points:

$$\begin{pmatrix} \theta - \Delta\theta \\ I(\theta - \Delta\theta, \rho) \end{pmatrix}, \quad \begin{pmatrix} \theta \\ I(\theta, \rho) \end{pmatrix}, \quad \begin{pmatrix} \theta + \Delta\theta \\ I(\theta + \Delta\theta, \rho) \end{pmatrix}$$

2. The same procedure is repeated for ρ , fitting a quadratic $Q_\rho(\rho)$ to the image peak and its two vertically adjacent points

The locations of the maxima of Q_θ and Q_ρ are the θ and ρ coordinates of the peak in the Radon image. Peaks detected in the Radon image and the corresponding image lines are depicted in Fig. 4.7.

4.2.1.2 Finding sets of parallel lines by robust vanishing point detection

The images of lines that are parallel in space always meet at a vanishing point. The lines detected on the cube are in one of three directions. Since most of the lines detected by the Radon transform meet at one of three vanishing points the lines detected by the Radon transform procedure can be partitioned into three sets by finding the three associated vanishing points.

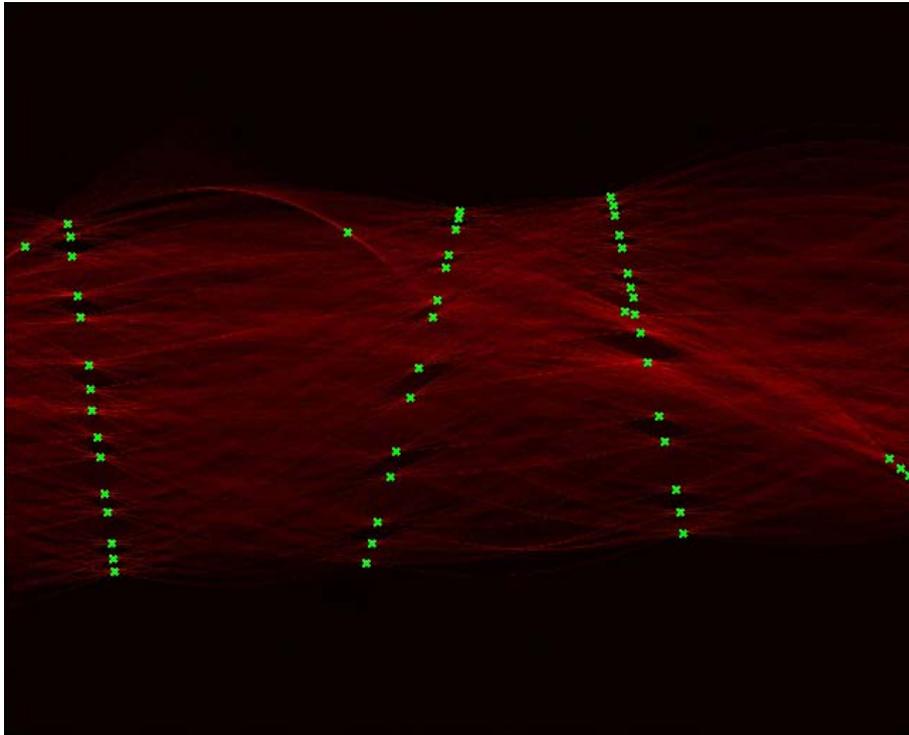
The RANSAC algorithm [12] is designed to robustly estimate a model from a set of points, and RANSAC thus incorporates outlier rejection. To find the three sets of image lines corresponding to parallel lines on the cube, a modified version of RANSAC was developed for detecting multiple vanishing points, Multiple-Model RANSAC (MMRANSAC), and is presented in Algorithm 4.3.

MMRANSAC was used to robustly detect vanishing points using parameters $N = 500$, $d_v = 1000$ pixels, and $d_{in} = 50$ pixels. The three detected vanishing points and corresponding three groups of parallel cube lines are shown in Fig. 4.8. The number of iterations N is much higher than required to guarantee, with high probability, that each group of lines will be found. The reason for choosing such a high N was simply that the MMRANSAC stage of the calibration algorithm takes very little time compared with the image processing stages, and therefore it does not degrade overall computational performance to use such a high value of N .

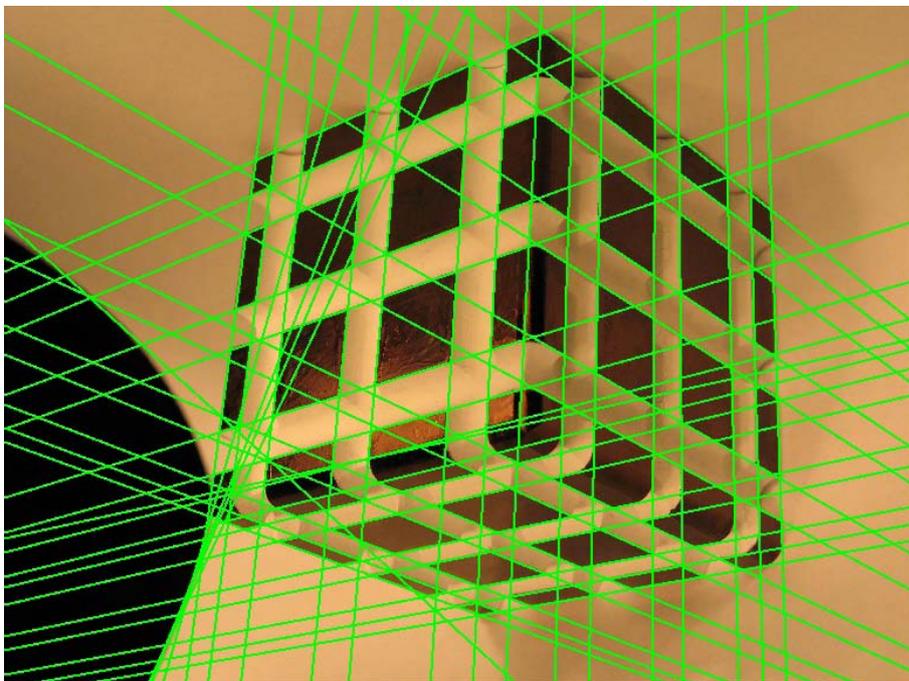
4.2.1.3 Vertex detection

If two of the groups of lines corresponding to two vanishing points are intersected pairwise, the resulting set of points contains all of the desired vertices from one face. For example, in Fig. 4.8, if all of the (blue) lines corresponding to vanishing point \mathbf{v}_2 are intersected with all of the (red) lines corresponding to \mathbf{v}_1 , then the resulting set of intersections contains the face of the cube in the top left of the image, as shown in Fig. 4.8b. It can also be seen that the lines that, when intersected, generate the desired points are the lines closest to the line joining \mathbf{v}_1 and \mathbf{v}_2 , which is the *vanishing line* for the face.

In Fig. 4.8b, let the blue lines, $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{16}$, be ordered by angle from the vanishing line, closest first (top to bottom in the image), and the red lines $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_{15}$ be ordered similarly (left to right in the image). In this example, the set of lines which



(a) All detected Radon transform peaks. Note that although it appears that there are three sets of peaks aligned on a straight line, they actually lie on three sinusoids corresponding to vanishing points



(b) Corresponding image lines

Figure 4.7: All detected Radon transform peaks, and the corresponding image lines

Algorithm 4.3 MMRANSAC: An algorithm for robustly finding three vanishing points and associated image lines

Input:

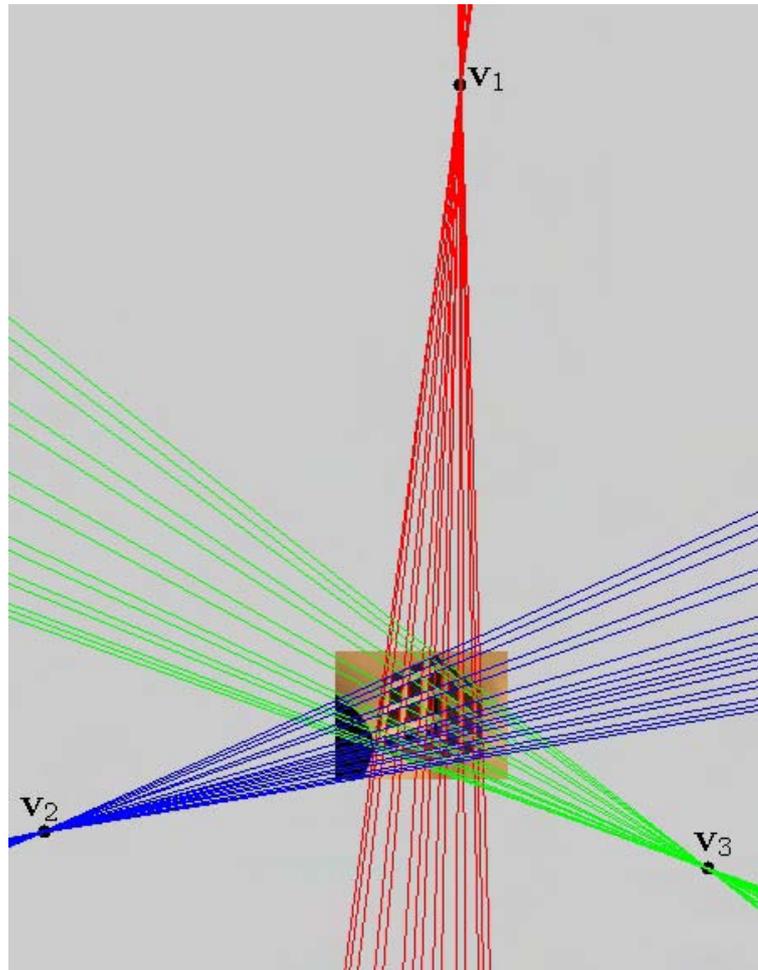
- A set \mathcal{L} of lines

Parameters:

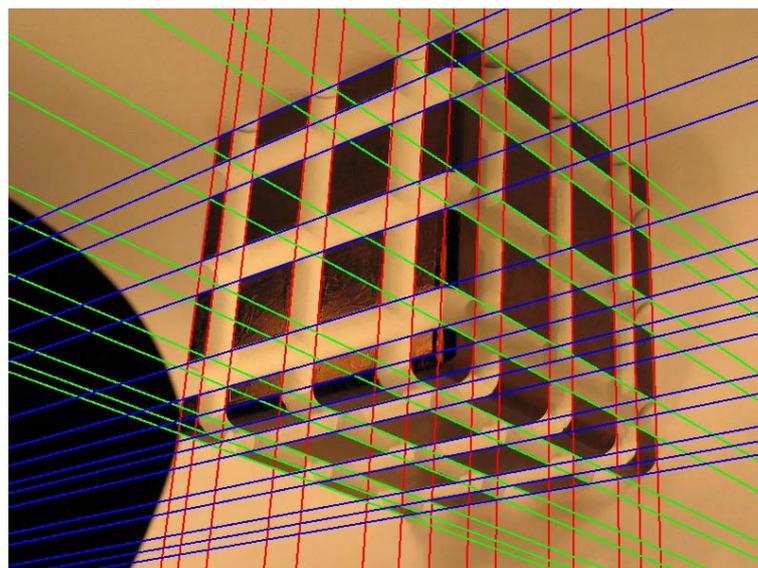
- d_v : The minimum distance in pixels that two distinct vanishing points must be apart to be detected
- d_{in} : The maximum perpendicular distance a line must be from a vanishing point to be classified as an inlier
- N : the number of iterations to perform

Algorithm:

1. Let \mathcal{V} be the set of detected vanishing points. Initialise $\mathcal{V} = \emptyset$
 2. Take a random sample of two lines from \mathcal{L} , and construct their intersection \mathbf{v}
 3. Construct the consensus set $S(\mathbf{v})$ of \mathbf{v} comprising all lines in \mathcal{L} that pass within d_{in} of \mathbf{v}
 4. **if** \mathcal{V} is empty
 - add \mathbf{v} to \mathcal{V}
 - else** find the nearest point $\mathbf{v}_n \in \mathcal{V}$ to \mathbf{v} (by Euclidean distance)
 - **if** $\|\mathbf{v}_n - \mathbf{v}\| < d_v$
 - if the consensus set of \mathbf{v} , $S(\mathbf{v})$, has more members than $S(\mathbf{v}_n)$, then replace \mathbf{v}_n with \mathbf{v} in \mathcal{V}
 - **elseif** there are fewer than three points in \mathcal{V}
 - add \mathbf{v} to \mathcal{V}
 - **else** (there are three points in \mathcal{V} , none of which are within d_v of \mathbf{v})
 - Find the point $\mathbf{v}_s \in \mathcal{V}$ that has the smallest consensus set $S(\mathbf{v}_s)$. In the case of a tie, choose one of the tied points at random
 - If $S(\mathbf{v})$ has more members than $S(\mathbf{v}_s)$, replace \mathbf{v}_s with \mathbf{v} in \mathcal{V}
 5. Repeat from step 2 until N iterations have been performed
 6. For each of the three vanishing points $\mathbf{v}_i \in \mathcal{V}$, reestimate \mathbf{v}_i by least squares from *all* the lines in its consensus set $S(\mathbf{v}_i)$. The consensus set is the set of inlying lines to \mathbf{v}_i
-



(a) Detected vanishing points v_1 , v_2 , v_3 , and the groups of inlying lines for each



(b) Detected parallel lines on image

Figure 4.8: Detected vanishing points and inliers from MMRANSAC algorithm for vanishing point estimation

generate the desired 36 vertices are $\mathbf{b}_2, \mathbf{b}_3, \dots, \mathbf{b}_7$ and $\mathbf{r}_2, \mathbf{r}_3, \dots, \mathbf{r}_7$. The intersections are stored in a 6×6 array $\mathcal{X}^{(1,2)}$, ordered as follows:

$$\mathcal{X}^{(1,2)} = \begin{array}{c|cccccc} & \mathbf{b}_2 & \mathbf{b}_3 & \mathbf{b}_4 & \mathbf{b}_5 & \mathbf{b}_6 & \mathbf{b}_7 \\ \hline \mathbf{r}_2 & \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \mathbf{x}_{1,3} & \mathbf{x}_{1,4} & \mathbf{x}_{1,5} & \mathbf{x}_{1,6} \\ \mathbf{r}_3 & \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \mathbf{x}_{2,3} & \mathbf{x}_{2,4} & \mathbf{x}_{2,5} & \mathbf{x}_{2,6} \\ \mathbf{r}_4 & \mathbf{x}_{3,1} & \mathbf{x}_{3,2} & \mathbf{x}_{3,3} & \mathbf{x}_{3,4} & \mathbf{x}_{3,5} & \mathbf{x}_{3,6} \\ \mathbf{r}_5 & \mathbf{x}_{4,1} & \mathbf{x}_{4,2} & \mathbf{x}_{4,3} & \mathbf{x}_{4,4} & \mathbf{x}_{4,5} & \mathbf{x}_{4,6} \\ \mathbf{r}_6 & \mathbf{x}_{5,1} & \mathbf{x}_{5,2} & \mathbf{x}_{5,3} & \mathbf{x}_{5,4} & \mathbf{x}_{5,5} & \mathbf{x}_{5,6} \\ \mathbf{r}_7 & \mathbf{x}_{6,1} & \mathbf{x}_{6,2} & \mathbf{x}_{6,3} & \mathbf{x}_{6,4} & \mathbf{x}_{6,5} & \mathbf{x}_{6,6} \end{array} \quad (4.1)$$

where the $(1, 2)$ superscript refers to the vanishing points $\mathbf{v}_1, \mathbf{v}_2$. The red lines are chosen to correspond to rows of the array and the blue points to columns. The six points on the first row of $\mathcal{X}^{(1,2)}$ are on the perimeter of the set of 36 vertices for the face, starting at the farthest point from the central vertex, and moving anticlockwise about the perimeter. Therefore, this organisation is a well-defined ordering that is independent of the order of the vanishing points chosen.

This array ordering can be achieved, in general, by choosing the set of lines that move outward *clockwise* from the vanishing line to correspond to the *rows* of the array, and the set of lines that move outward *anticlockwise* from the vanishing line to correspond to *columns* of the array. Referring to Fig. 4.8a, it can be seen that the choice of ordering for the red and blue lines is consistent with this definition, and that it also holds for the other two pairs (red-green and green-blue).

Although in the example of Fig. 4.8 it was easily seen that the desired set of lines for each face were the 2nd to 7th of the lines ordered in ascending order of angle to the vanishing line, in general it is not that simple. Variations in lighting and camera pose can mean that the outside edge may not get measured, or that extra lines within the face are detected, due to shadows cast by the edges of the cube. Therefore, it is necessary to devise a test to determine which six lines are the correct six.

For one of the vanishing points \mathbf{v} for a specific face, the six lines being sought are coplanar. Since the six lines also meet at a common point, \mathbf{v} , ordered groups of four lines from the six have a cross ratio invariant (§3.2.5.4). Further, the image coordinates for the face differ only by a projective transformation from any 2D Cartesian coordinates for the world face. Therefore, it is possible to find a projective transformation that intersects the six lines with the following colinear points: $\mathbf{x}_1, \dots, \mathbf{x}_6$:

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 6 \\ 1 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 18 \\ 1 \end{pmatrix} \quad \mathbf{x}_4 = \begin{pmatrix} 24 \\ 1 \end{pmatrix} \quad \mathbf{x}_5 = \begin{pmatrix} 36 \\ 1 \end{pmatrix} \quad \mathbf{x}_6 = \begin{pmatrix} 42 \\ 1 \end{pmatrix}.$$

The numbers are the spacings (in mm) of the desired interest points on the cube. Three

cross ratios can be computed for the six ordered points, and these are:

$$\begin{aligned} C_1 &= \text{Cross}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) \\ C_2 &= \text{Cross}(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) \\ C_3 &= \text{Cross}(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6) \end{aligned} \tag{4.2}$$

The three cross ratios of the intersections of the correct six lines (in the correct order) with *any* line will yield the same three invariants (C_1, C_2, C_3) . For the six points above, the cross ratios are $(C_1, C_2, C_3) = (\frac{1}{9}, \frac{4}{9}, \frac{1}{9})$. The following procedure can therefore be used to identify the correct six lines:

1. Construct the perpendicular bisector \mathbf{l}_\perp to the vanishing line \mathbf{l}_v
2. Compute the intersections of all of the lines emanating from the vanishing point \mathbf{v}_i with \mathbf{l}_\perp
3. Sort the intersections according to distance from \mathbf{l}_v in ascending order
4. Discard the farthest six lines from \mathbf{l}_v . If any of these were chosen, it would not be possible to find six lines to compute intersections for the other face with lines from \mathbf{v}
5. From the remaining n points (there will be at most around 10), choose all $\binom{n}{6}$ combinations of six points, and compute the three cross ratios (4.2) for each combination
6. Choose the configuration of six points that is closest in the least squares sense to $(\frac{1}{9}, \frac{4}{9}, \frac{1}{9})$ as the correct ordered set of six points

The vertices detected in this manner for the example cube are shown in Fig. 4.9

Once the three sets of features $\mathcal{X}^{(1,2)}$, $\mathcal{X}^{(1,3)}$, $\mathcal{X}^{(2,3)}$ have been found, the remaining problem before camera resection can be formed is feature correspondence – matching the sets of features to the world faces of the cube.

4.2.2 Feature correspondence

Because all of the faces of the cube are the same, there is no way of distinguishing which faces are visible without user input. Therefore, once the features are detected, the user is asked to identify each of the three faces as top, bottom, left, right, front, or back. The actual identities are not important as long as the user is consistent in the choices from each camera, so that a face identified as the ‘top’ face from one camera must also be denoted as the ‘top’ face from another camera.

The GUI that the user is presented with to do this task is shown in Fig. 4.10. In this particular example, the blue face has been identified as the ‘bottom’ face by the

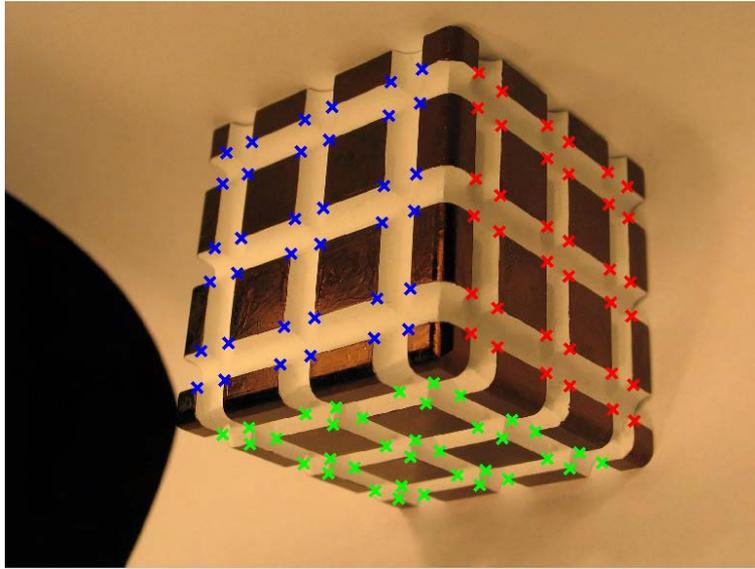


Figure 4.9: Detected vertices, organised by face

user, the red face as the ‘front’ face, and the green face as the ‘right’ face. However, it is clear that this portion alone requires significant input, costs some extra time, and is not robust to user error or misuse. In particular, correctly identifying the same faces from different cameras is a potentially difficult and confusing task.

Because of the ordering of the image points, and the grouping into faces, the final resulting task of identifying features with their world coordinates is simple. The image coordinates are ordered as in (4.1), with the nearest point to the centre at the bottom right of the array, with the points in the first row of the array identifying the anticlockwise direction around the perimeter of the set of 36 points.

The coordinates of the world faces are stored in six 6×6 arrays with the same condition that the first row corresponds to the anticlockwise direction around the face. To match the features between the identified image faces and the corresponding world faces, the world arrays themselves, not the contained coordinates, are rotated about their centre so that the vertex nearest the cube corner in common to the three faces is in the bottom-right position. The entries of the world array then match directly with the corresponding image array entries. Camera calibration can then proceed by resection as described in §3.5.

4.3 Experimental Results

The algorithm was implemented entirely in Matlab. Two cameras were calibrated from two views of the cube, with images as shown in Fig. 4.11. The cameras were positioned so three cube faces were clearly visible from both cameras. To obtain sufficient accuracy

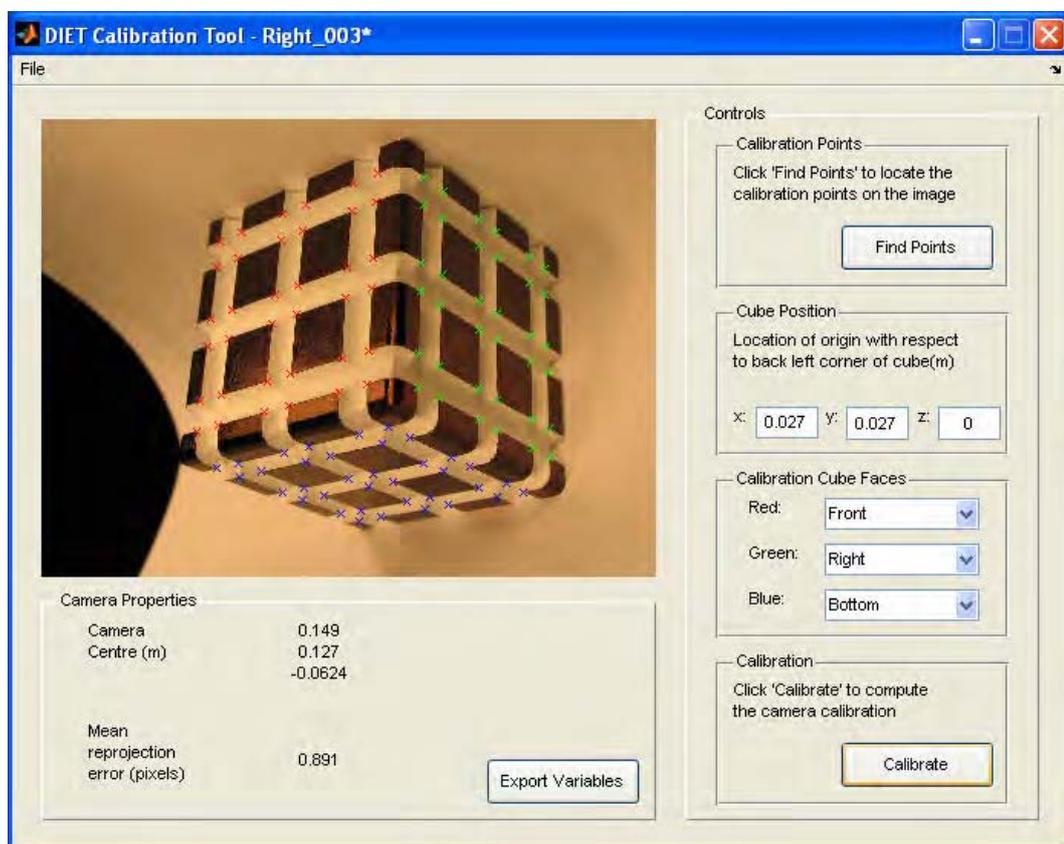


Figure 4.10: Calibration user interface

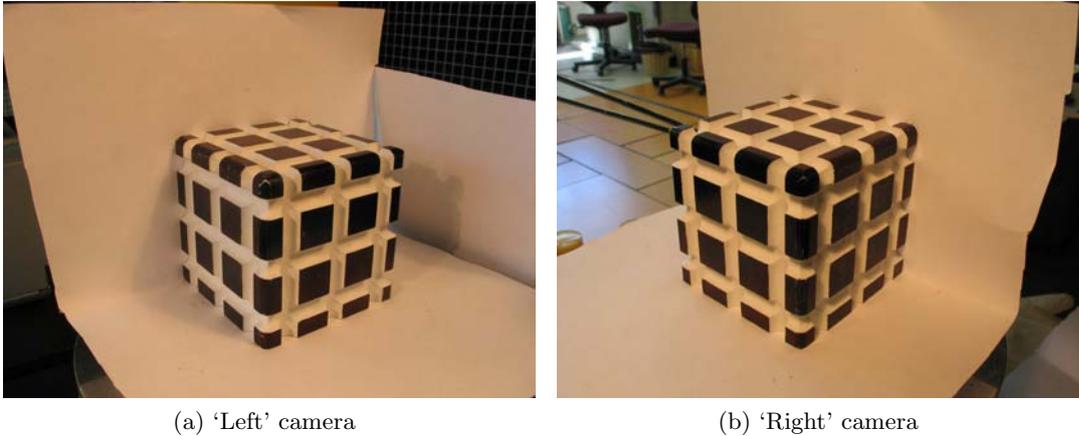


Figure 4.11: The two views of the cube used to calibrate the two cameras

Camera	Position (mm)	Calibration matrix K_i
Left	$\begin{pmatrix} 133.5 \\ -98.8 \\ 97.2 \end{pmatrix}$	$\begin{bmatrix} 2580 & 3.11 & 1336.4 \\ 0 & 2574 & 993.3 \\ 0 & 0 & 1 \end{bmatrix}$
Right	$\begin{pmatrix} 114.1 \\ 100.7 \\ 103.3 \end{pmatrix}$	$\begin{bmatrix} 2580 & 3.32 & 1213.4 \\ 0 & 2462 & 956.8 \\ 0 & 0 & 1 \end{bmatrix}$

Table 4.1: Computed 3D coordinates of the optical centre and the intrinsic calibration matrix for each camera

of the line measurements, the cameras were run at 5.0 megapixels (2592×1944 pixels).

As a small test for robustness, the image backgrounds were not entirely masked, so numerous additional straight lines are present in the images. Two of the faces are visible from both cameras, the 'top' and 'front' faces. The world coordinate system was oriented with the origin at the center of the bottom of the cube, and the z -axis pointing directly upward.

The cameras were calibrated with the algorithm described in this chapter. Calibration of each camera took approximately two minutes on a 1.6GHz Pentium M laptop, with almost all the time being dedicated to the Radon transform and peak finding components of the algorithm. The position and orientation of the two cameras with respect to the cube, as computed by camera resection is shown in Fig. 4.12. The computed intrinsic calibration matrices K_i and camera centres are shown in Table 4.1.

An experiment was performed to measure the accuracy of the calibration method. The measured image coordinates of the two overlapping faces of the cube from both cameras were used to reconstruct features in 3D by triangulation (§3.6.3). The differ-

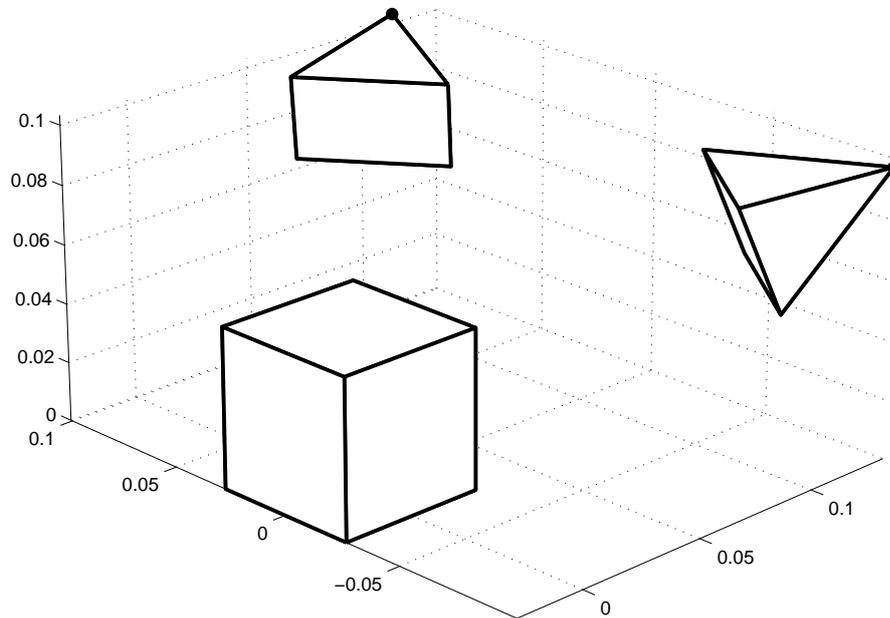


Figure 4.12: Computed position and orientation of the two cameras, with respect to the cube

ence between the reconstructed 3D coordinates and the ground truth was computed to assess accuracy.

72 points were reconstructed in 3D, comprising the top and front faces of the cube. The mean error from ground truth was 0.093mm with standard deviation 0.044mm and the maximum error was 0.22mm. The reconstructed points are shown as blue dots in Fig. 4.13.

4.4 Discussion and Conclusion

The experiments performed in §4.3 show that this method of calibrating the cameras is very accurate. An average reconstruction error of 0.1mm is at the same level as the manufacturing tolerance of the calibration cube. However, this accuracy comes at a price. For the lines to be detected sufficiently accurately, high resolution images of 5 megapixels were required. For lower resolution images, the level of error in the line measurements resulted in significant errors in the cross ratio calculations and, as a result, wrong sets of six lines being chosen. The Radon transform is very slow, having to do a complex calculation for every pixel in the parameter space. As a result, calibration takes a long time. This problem is compounded if calibration does not work, and the user has to adjust the system.

It was found that the algorithm only worked when three faces of the cube were very clearly visible, such as in the scenario depicted in Fig. 4.11. As soon as one face was

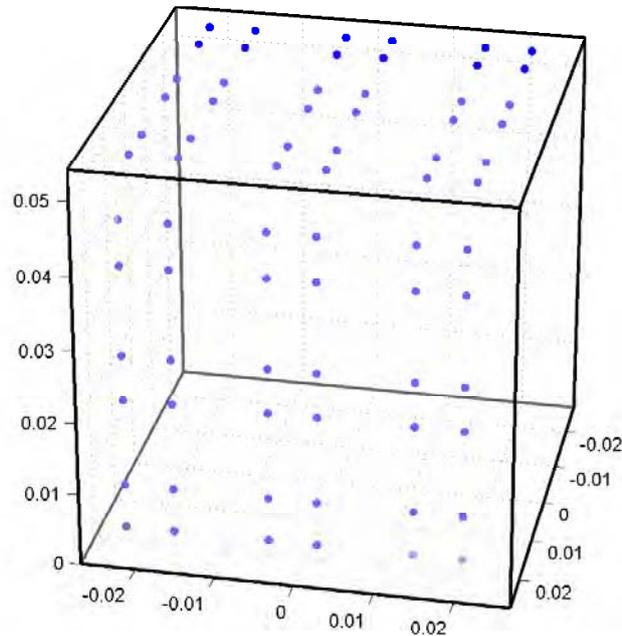


Figure 4.13: Reconstructed features from two views (blue dots - the cube boundary is shown for context)

only obliquely visible, the edge detection routine had difficulty recognising the edges. Finally, user input is required in all cases, adding time and potential error.

The user input is minimal, and simple for the case where only two cameras are used. However, for a larger array of cameras, it becomes less clear which faces of the cube are present, and the likelihood of human error increases significantly. Moreover, the results of human error in calibration will not necessarily be detected until after the 3D breast reconstruction has been attempted, leading to further practical issues.

Therefore, in spite of the accuracy of the calibration method, the drawbacks were sufficiently high that a new method was required to overcome these shortcomings. This novel approach and method is presented in Chapter 5.

Chapter 5

SILHOUETTE-BASED CALIBRATION

5.1 Introduction and Overview

This chapter describes a method for fully automatically calibrating a camera from a single image of a cubic calibration object. The cubic calibration object has circular features on each face, arranged in different patterns on each face to enable each face to be uniquely identified. The calibration method described lends itself to situations requiring fast calibration of numerous cameras or unattended calibration because only a single image from each camera is required, with no user input.

The novelty of the method is the use of homographies (projective transformations) between each of the two or three visible calibration cube faces and a reference square in \mathbb{R}^2 . The problems of face identification and world-image feature matching are performed in this reference square. The homographies are determined from the cube face boundaries in the images, which, in turn, are identified from the silhouette of the cube alone. Because only the silhouette is used to compute the homographies, the procedure can work effectively in a wide variety of imaging conditions. The only image processing required is a means of segmenting the cube from the background and segmenting the features from the cube.

In addition to facilitating the matching process, the computed homographies have a significant benefit in accurate feature measurement. The circular features of the cube are imaged as ellipses. The computed homographies remove perspective distortion from the images of the cube faces, mapping the ellipses back to circles. The centres of the circles can be used as unbiased measurements of the centres of the cube circle centres, and the circle centres can be measured to subpixel precision in this way.

It is not sufficient to simply compute the centres of the measured ellipses, as these are biased by perspective distortion. Typically, this distortion is compensated for by an iterative procedure after calibration has been computed [17] or not at all. In contrast,

this method allows the distortion to be compensated for before the calibration is computed, significantly simplifying the procedure. Note that circular features are desirable, compared with often-used corner, checkerboard and line intersection features, as the measurement of the centres does not suffer the same sensitivity to image thresholding procedures.

Once the feature correspondences have been established, the maximum likelihood estimate of the camera model (assuming Gaussian noise) is computed by camera resection, as described in §3.5.1.

5.2 Method

5.2.1 Algorithm overview

The calibration algorithm developed in this chapter consists of the fundamental steps presented in Algorithm 5.1. Each of these steps will be presented in the subsequent sections, with reference to a simple calibration object to illustrate and construct the procedure.

Algorithm 5.1 High level calibration algorithm

1. Extract the silhouette of the cube from the background
 2. Extract elliptical regions from the image, and fit ellipses to their boundaries.
 3. Find the edges of the cube from the silhouette boundary, and construct the face boundaries of the two or three visible faces
 4. For each face:
 - (a) Compute a homography mapping the four corners of the face to the four corners of the unit square and use this to map the elliptical features within the face to circles inside the square
 - (b) Compute the centres of the circles to use as image point measurements
 - (c) Deduce which of the six faces is under consideration by rotating the image face coordinates in the reference square in 90° steps and matching each rotated version to the six templates representing the faces of the cube
 5. By using knowledge of adjacent faces, align the image and world projections in the reference square, allowing point correspondences to be found by nearest neighbour matching
 6. Compute the camera calibration by camera resection
-

The calibration object is depicted in Fig. 5.1, showing images of the cube from two different positions with two and three faces of the cube visible. These two pictures represent two fundamentally different configurations. In panel (b), three faces of the

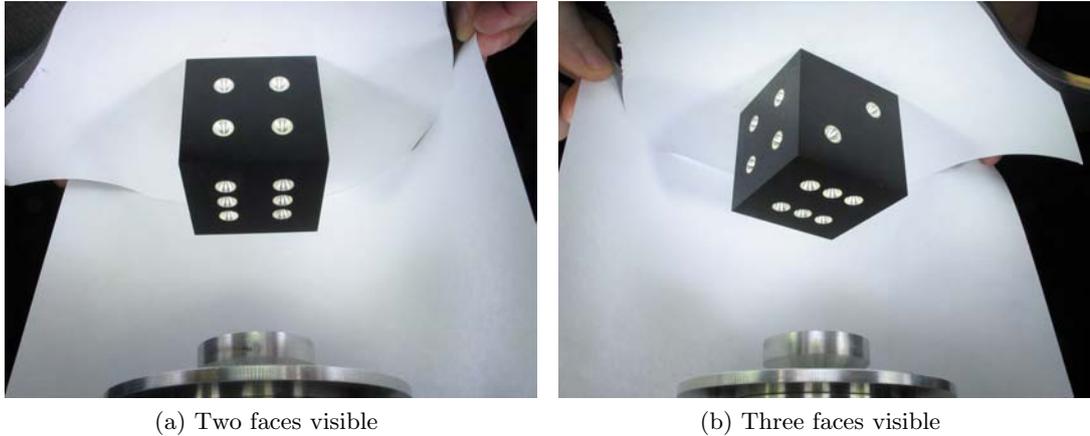


Figure 5.1: Two different views of the calibration cube used to illustrate the calibration procedure. Note that the cube is easy to segment from the background but that the interior edges are indistinct and therefore difficult to determine by standard image processing techniques

cube are visible, whereas in panel (a) only two faces are visible. There are a small number of degenerate viewing angles possible, for example where only one face is visible, which cause the algorithm to fail. These cases will be discussed in §5.2.3.3.

5.2.2 Identifying the cube silhouette and elliptical features

The calibration cube is designed so that it is readily segmented from the background by simple thresholding. The procedure for finding the silhouette of the cube and the elliptical features is illustrated in Fig. 5.2 for the cube from Fig. 5.1b. The grayscale image is first thresholded as depicted in panel (a). A connected component analysis is performed on the resulting binary image. The cube, \mathcal{I}_c , is then found by identifying the connected component, larger than a minimum size, closest to the centre of the image, as shown in panel (b). The cube silhouette, \mathcal{I}_s , is then found by morphological filling of \mathcal{I}_c , as shown in panel (c). The dot regions, \mathcal{I}_d , are found by performing a logical AND on the cube silhouette, \mathcal{I}_s , and the complement of \mathcal{I}_c , as shown in panel (d). The six exterior silhouette edges of the cube are found by fitting line segments to the convex hull of the silhouette, as shown in panel (e). Finally, a connected component analysis is performed on the dot region image and ellipses are fitted to the convex hull of each region, as shown in panel (f). The ellipses are fitted by nonlinear least squares to minimise geometric error [13]. Once the silhouette and ellipses are found, the image-processing component of the calibration algorithm is complete.

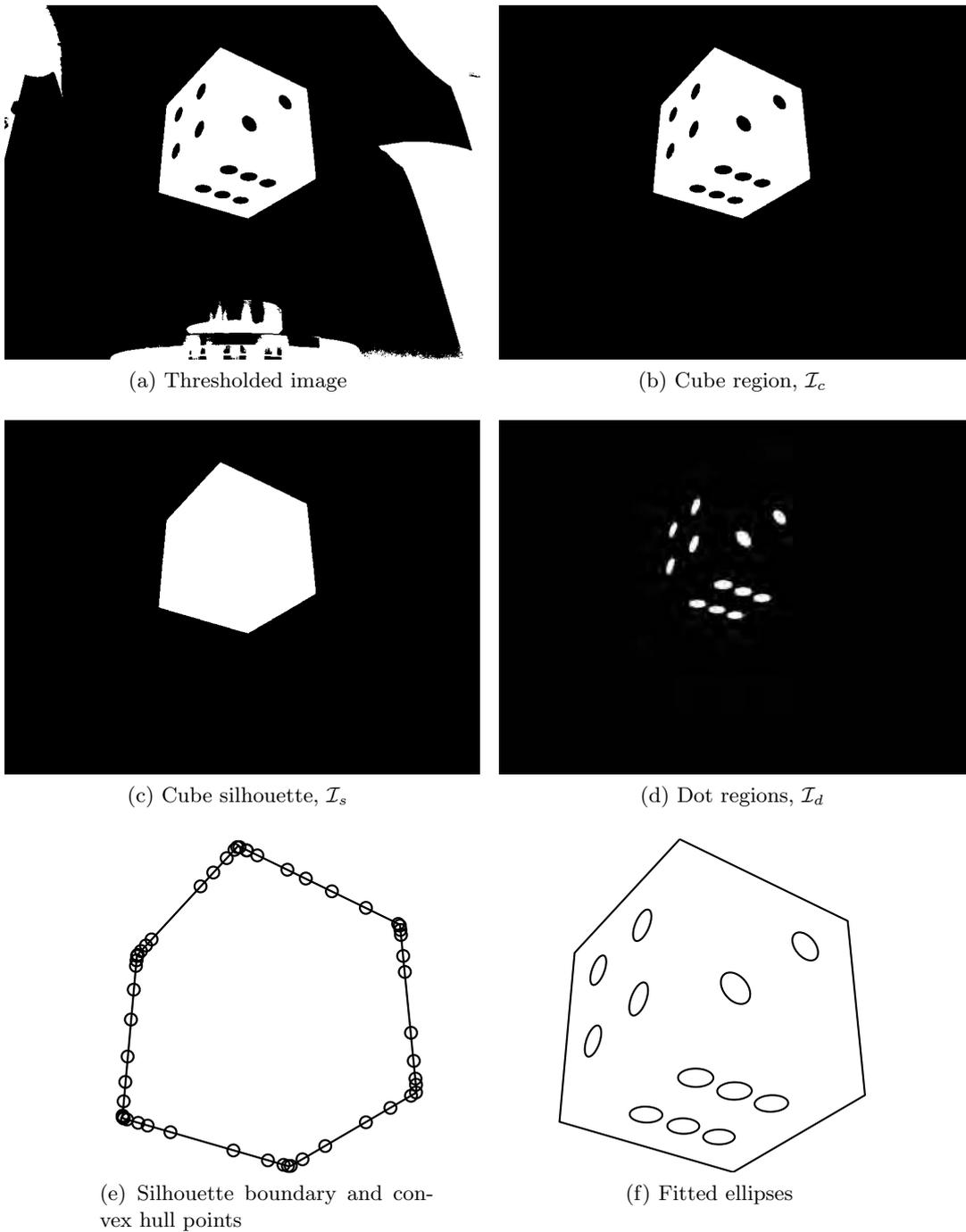


Figure 5.2: Identifying the silhouette and elliptical features for the cube depicted in Fig. 5.1b

5.2.3 Partitioning the image into cube faces

The procedure for partitioning the cube into two faces differs depending on whether two or three faces of the cube are visible. If only two faces are visible, there is a condition which must be met by the six vertices of the silhouette. If this condition, discussed in §5.2.3.1, is not met, then it is assumed that there are three faces visible and the partitioning procedure presented in §5.2.3.2 is used.

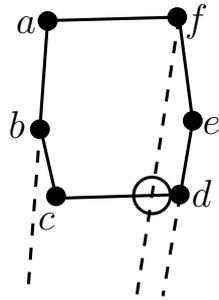
5.2.3.1 Partitioning the cube into two faces

Consider the silhouette of the cube shown in Fig. 5.1a. The vertices can be labelled anticlockwise (with no preferred choice of start point) a, \dots, f as shown in Fig. 5.3. One pair of opposite edges of the silhouette corresponds to parallel cube edges, the other two correspond to skew edges. Consider a pair of opposite silhouette edges, for example the line segments $\overline{ab}, \overline{ed}$. If these two lines are the pair corresponding to the parallel cube edges, then the line joining c and f will correspond to another parallel cube edge. The lines containing $\overline{ab}, \overline{ed}$ and \overline{cf} should therefore meet at a single vanishing point. This set of criteria can be tested by the following procedure, as shown in Fig. 5.3a:

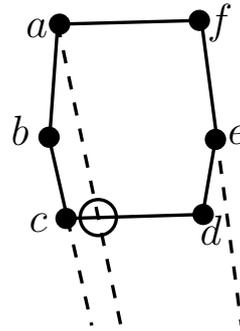
1. Intersect the lines containing \overline{ab} and \overline{ed} in a point v (note v is not shown in Fig. 5.3a because it is a long way off the image)
2. Construct the line joining v to f . The vertex f is chosen because it is the farthest remaining vertex from v . The constructed line shown as the dashed line emanating from f
3. If \overline{ab} and \overline{ed} are the pair corresponding to parallel edges, then the constructed line should pass through c

The constructed line does not pass through d , and therefore \overline{ab} and \overline{ed} are not the images of parallel edges. The procedure is then repeated with the next pair of edges $\overline{bc}, \overline{ef}$. A vanishing point v is constructed and joined by a line to a , as shown in Fig. 5.3b. The constructed line does not pass through d , hence \overline{bc} and \overline{ef} are not the parallel pair. The procedure is then repeated for the final pair $\overline{cd}, \overline{fa}$. This time, the vanishing point v is constructed, v is joined to b , and the resulting line passes through the remaining vertex, e . Therefore, \overline{cd} and \overline{fa} correspond to parallel edges of the cube, and the interior edge of the cube is the line segment \overline{be} . Referring to Figures. 5.1 and 5.3d, the ‘4’ face of the cube has vertices (b, e, f, a) and the ‘6’ face of the cube has vertices (e, b, c, d) . By convention, the vertices are ordered anticlockwise with the two overlapping vertices b, e listed first.

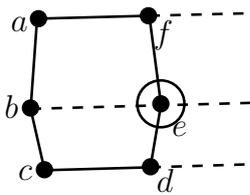
If there are three faces of the cube visible, then every pair of opposite sides correspond to parallel cube edges, as can be seen in Fig. 5.1b. However, for a given pair of opposite edges, the line in space corresponding to the line joining the remaining



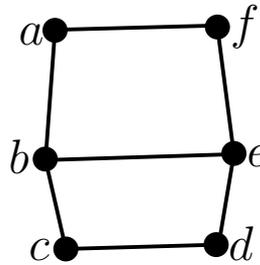
(a) \overline{ab} , \overline{de} are not parallel edges of the cube as the constructed line doesn't pass through c



(b) \overline{bc} , \overline{ef} are not parallel edges of the cube as the constructed line doesn't pass through e



(c) \overline{cd} , \overline{fa} are parallel edges of the cube as the constructed line passes through e



(d) The two faces have corners (b, e, f, a) and (e, b, c, d)

Figure 5.3: Partitioning the cube into two faces for the two faces visible case

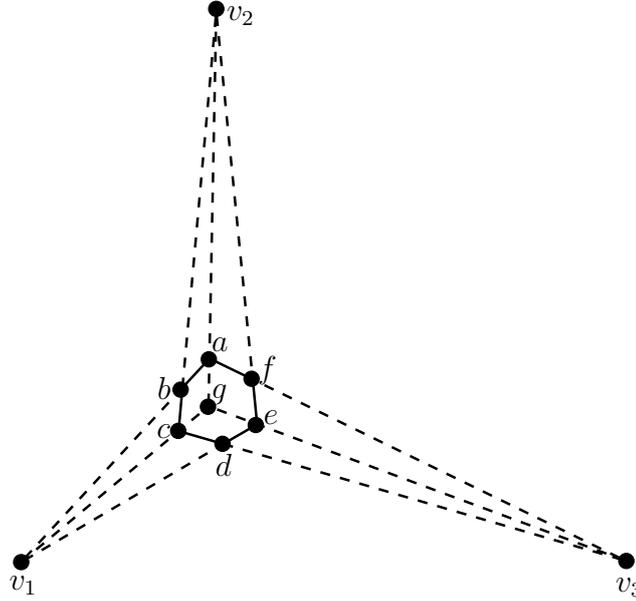


Figure 5.4: The vanishing point v_1 is constructed by intersecting the lines containing \overline{ab} and \overline{de} . The closer of the two remaining vertices (c and f) to v_1 is vertex c , so the line passing through v_1 and c will pass along one of the cube edges, through the centre vertex. The same procedure is followed to construct v_2 and v_3 and the other two lines passing through the centre. The centre, g , is computed as the intersection of the three lines by least squares. The boundaries of the three faces of the cube are therefore (g, a, b, c) , (g, c, d, e) , (g, e, f, a) , ordered anticlockwise from g by convention

two vertices is not parallel to the edge pair. No pair of sides will therefore satisfy the parallel condition. Therefore, if the partitioning procedure for two faces fails, it is assumed that three faces are visible, and the three-face partitioning procedure described in §5.2.3.2 is performed.

5.2.3.2 Partitioning the cube into three faces

If there are three faces of the cube visible, such as can be seen in Fig. 5.1b, then before the cube can be partitioned, the vertex at the intersection of the three faces must be constructed. Label the silhouette vertices anticlockwise as a, \dots, f . Each pair of opposite silhouette edges correspond to parallel edges of the cube. The vanishing point for the direction can therefore be constructed by intersecting the lines containing the opposite edges. Lines containing the remaining three edges can be constructed by joining each vanishing point to the closer of the two vertices not used to construct it. The procedure for computing these lines for the silhouette of the cube in Fig. 5.1b is shown in Fig. 5.4. The intersection of these three lines gives the central vertex g .

Let the three lines be written as $\alpha_i x + \beta_i y + \gamma_i = 0$ for $i = 1, \dots, 3$ where $\alpha_i^2 + \beta_i^2 = 1$. Let g have coordinates $(x_g, y_g)^\top$. The centre vertex, g , can then be found by finding

the least squares solution to the linear matrix equation

$$\begin{bmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \alpha_3 & \beta_3 \end{bmatrix} \begin{pmatrix} x_g \\ y_g \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} \quad (5.1)$$

This formulation minimises the sum of squares of the perpendicular distance from g to each line. Once g has been computed, the three faces of the cube are known. Referring to Fig. 5.4 and Fig. 5.1, the ‘4’ face of the cube has vertices (g, a, b, c) , the ‘6’ face of the cube has vertices (g, c, d, e) and the ‘2’ face of the cube has vertices (g, e, f, a) . By convention, the vertices are ordered anticlockwise starting from g .

5.2.3.3 Degenerate viewing angles

There are a small number of degenerate configurations which will cause the partitioning algorithm to fail. These are depicted in Fig. 5.5. The case represented by Fig. 5.5a fails because it is not possible to identify the six exterior vertices on the silhouette, and hence not possible to construct the interior edge line and partition the cube. Fig. 5.5b, where only one face is visible fails, obviously, and Fig. 5.5c fails because one pair of sides satisfies the parallel condition testing for two faces, and hence will be partitioned incorrectly into two faces. Viewpoints near (a) and (c) are the most likely to arise in practice, but can generally be avoided by a small rotation of the cube. Case (c) can also be mitigated by forcing the algorithm to perform the three face partitioning algorithm, skipping the test for two faces.

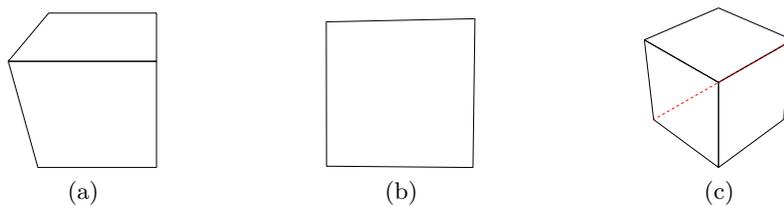


Figure 5.5: Degenerate viewing angles

5.2.4 Finding point correspondences

The key to finding the correspondences between the world features on the cube and the measured image features is to find transformations mapping each measured face and each cube face to the same two-dimensional space. These transformations can be determined in a way such that the four corners of each face are mapped to the four corners of the unit square in \mathbb{R}^2 . Moreover, it is possible to compute the transformations in such a way that the orientation of each face is preserved.

A measured image face and its corresponding cube face will have the same feature locations (up to a noise threshold) in the square if the corners of the both faces have been mapped to the same corners of the unit square. If this situation is the case, the points can be matched by nearest neighbour matching. If corresponding corners have not been mapped to the same corners of the square, then, because the transformations are orientation-preserving, the two sets of coordinates differ by a rotation of $\pi/2$, π , or $3\pi/2$ about the centre of the square.

It is not possible to guarantee *a priori* that the corners of the imaged faces will be mapped to the same corners of the square as the cube faces. Therefore, each of the four possible orientations of the image face must be considered when matching against each of the six world faces to determine the face identities. The face identification procedure thus involves twenty-four comparisons for each measured face.

5.2.4.1 Mapping image faces to the unit square

In the face-partitioning procedure, the four corners of two or three faces of the cube were identified. For each of these faces, there exists a homography (projective transformation) mapping the face to the unit square. Under this transformation, all projective distortion introduced by the imaging process is removed. In particular, the ellipses are mapped back to circles. The homography can be computed by specifying which of the four corners of the square each corner of the face is mapped to. It is important to preserve the orientation of the face by ensuring the anticlockwise order of the points is preserved. There are therefore four ways to compute the homography. By convention, the four vertices of the image face are ordered as in §5.2.3.1 and §5.2.3.2, and Figures 5.3 and 5.4, and the corresponding corners of the unit cube are ordered anticlockwise from the origin.

In homogeneous coordinates, the homography is represented by an invertible matrix $\mathbf{H} \in \mathbb{R}^{3 \times 3}$, which is unique up to a scale factor. The action of the homography on a point $\mathbf{x} = (x, y, 1)^\top$ is given by matrix multiplication as:

$$\lambda \bar{\mathbf{x}} = \mathbf{H}\mathbf{x}, \quad (5.2)$$

where $\bar{\mathbf{x}} = (\bar{x}, \bar{y}, 1)^\top$ are the coordinates of the mapped point, and λ is a nonzero scalar. For a given correspondence $\mathbf{x} \Leftrightarrow \bar{\mathbf{x}}$, the quantities $\mathbf{H}\mathbf{x}$ and $\bar{\mathbf{x}}$ are related by scalar multiplication (by λ), so a constraint on \mathbf{H} , not involving λ can be formed by taking the cross product:

$$\bar{\mathbf{x}} \times \mathbf{H}\mathbf{x} = \mathbf{0}. \quad (5.3)$$

The cross product (5.3) provides three linear constraints on the entries of \mathbf{H} , two of which are linearly independent. Therefore, each correspondence contributes two constraints on the entries of \mathbf{H} . Because \mathbf{H} is defined only up to scale, four correspondences are

Vertex	Image coordinates (pixels)	Destination square corner
g	$\mathbf{x}_g = (721.1 \quad 465.8 \quad 1)^\top$	$\bar{\mathbf{x}}_g = (0 \quad 0 \quad 1)^\top$
a	$\mathbf{x}_a = (727.6 \quad 141.8 \quad 1)^\top$	$\bar{\mathbf{x}}_a = (1 \quad 0 \quad 1)^\top$
b	$\mathbf{x}_b = (545.8 \quad 339.9 \quad 1)^\top$	$\bar{\mathbf{x}}_b = (1 \quad 1 \quad 1)^\top$
c	$\mathbf{x}_c = (519.5 \quad 634.5 \quad 1)^\top$	$\bar{\mathbf{x}}_c = (0 \quad 1 \quad 1)^\top$

Table 5.1: Image coordinates and desired unit square coordinates of the corners of the ‘4’ face of the cube depicted in Fig. 5.1b. Note, the coordinates are homogeneous coordinates

required to completely determine \mathbf{H} . The scale is commonly determined by constraining \mathbf{H} to have unity Frobenius norm, or setting $h_{33} = 1$. For details on the computation of \mathbf{H} , see Hartley and Zisserman [16, Chapter 4]

An ellipse can be represented in homogeneous coordinates by a symmetric positive definite matrix $\mathbf{Q} \in \mathbb{R}^{3 \times 3}$, defined up to scale, as shown in §3.2.4. The equation for the ellipse is

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = 0. \quad (5.4)$$

The equation for the mapping of an ellipse \mathbf{Q} under the homography \mathbf{H} is given by (3.10):

$$\mathbf{Q} \mapsto \mathbf{H}^{-\top} \mathbf{Q} \mathbf{H}^{-1}. \quad (5.5)$$

An ellipse matrix \mathbf{Q} can be arbitrarily partitioned as follows:

$$\mathbf{Q} = \begin{bmatrix} \hat{\mathbf{Q}} & \mathbf{q} \\ \mathbf{q}^\top & q \end{bmatrix} \quad (5.6)$$

where $\hat{\mathbf{Q}} \in \mathbb{R}^{2 \times 2}$, $\mathbf{q} \in \mathbb{R}^2$, and $q \in \mathbb{R}$. The centre $\mathbf{x}_c = (x_c, y_c)^\top$ is then given [13] by solving the linear matrix equation

$$-\hat{\mathbf{Q}} \mathbf{x}_c = \mathbf{q}. \quad (5.7)$$

Consider the ‘4’ face of the cube from Figures 5.1b and 5.4. The four corners of this specific face in image space, ordered anticlockwise with the centre vertex first, are the vertices (g, a, b, c) in Fig. 5.4. The homogeneous image coordinates for each of these vertices and for the corresponding corners of the unit square are shown in Table 5.1. The homography \mathbf{H} mapping the coordinates as shown in Table 5.1 is computed from these correspondences, and the four ellipses are mapped as in (5.5).

The face before and after transformation is shown in Fig. 5.6. It should be noted that due to small errors in the fitted ellipses and face corner locations, the homography doesn’t completely remove the projective distortion in this case. The features still appear slightly elliptical. Small inaccuracies in the homography do not significantly

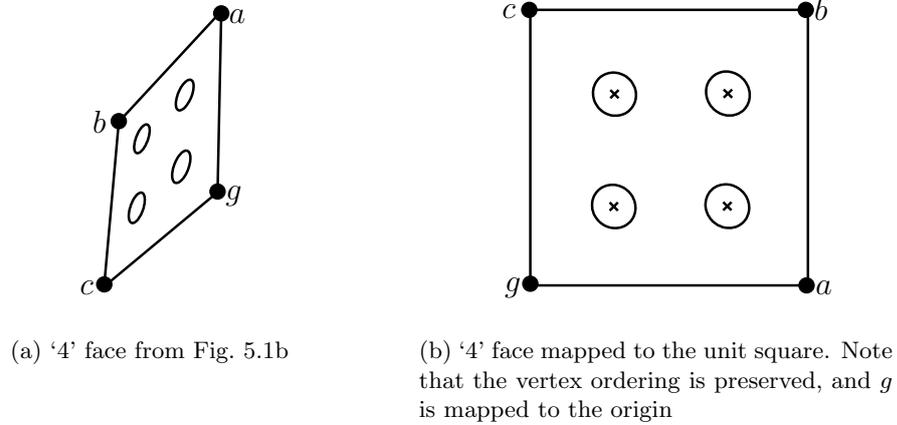


Figure 5.6: Mapping face '4', including the ellipses, from Fig. 5.1b to the unit square. Note that due to small inaccuracies in determining the ellipses and the corners of the face, the projective distortion is not completely removed, the features are still slightly elliptical after transformation.

face plane	die no.	mapping from $\mathbb{R}^3 \mapsto \mathbb{R}^2$	vertices (anticlockwise from origin)
$X = 0$	'5'	$(X, Y, Z) \mapsto (Z/h, Y/h)$	(1, 5, 7, 3)
$X = h$	'2'	$(X, Y, Z) \mapsto (Y/h, Z/h)$	(2, 4, 8, 6)
$Y = 0$	'3'	$(X, Y, Z) \mapsto (X/h, Z/h)$	(1, 2, 6, 5)
$Y = h$	'4'	$(X, Y, Z) \mapsto (Z/h, X/h)$	(3, 7, 8, 4)
$Z = 0$	'1'	$(X, Y, Z) \mapsto (Y/h, X/h)$	(1, 3, 4, 2)
$Z = h$	'6'	$(X, Y, Z) \mapsto (X/h, Y/h)$	(5, 6, 8, 7)

Table 5.2: Mappings into the unit square from each face of the cube

affect the accuracy of the calibration, as the coordinates in the reference square are only used for feature correspondence, and not in the computation of the projection matrix P . If the features appear only slightly eccentric, the difference between the ellipse centre and the circle centre will be negligible for most applications.

5.2.4.2 Mapping cube faces into the unit square

The cube, with side length h can be embedded into \mathbb{R}^3 as shown in Fig. 5.7. Each face of the cube has a natural mapping into the unit square in \mathbb{R}^2 which preserves the anticlockwise ordering of vertices (as observed from outside the cube). The mapping for each face is given in Table 5.2, together with the die numbers for the cube in Fig. 5.1 and the four vertices of the face, ordered anticlockwise from the origin of the unit cube.

The cube has side length $h = 54\text{mm}$. Consider again, for example, the '4' face of the cube. Reading from the fourth row of Table 5.2, the '4' face is on the plane $Y = h = 54$.

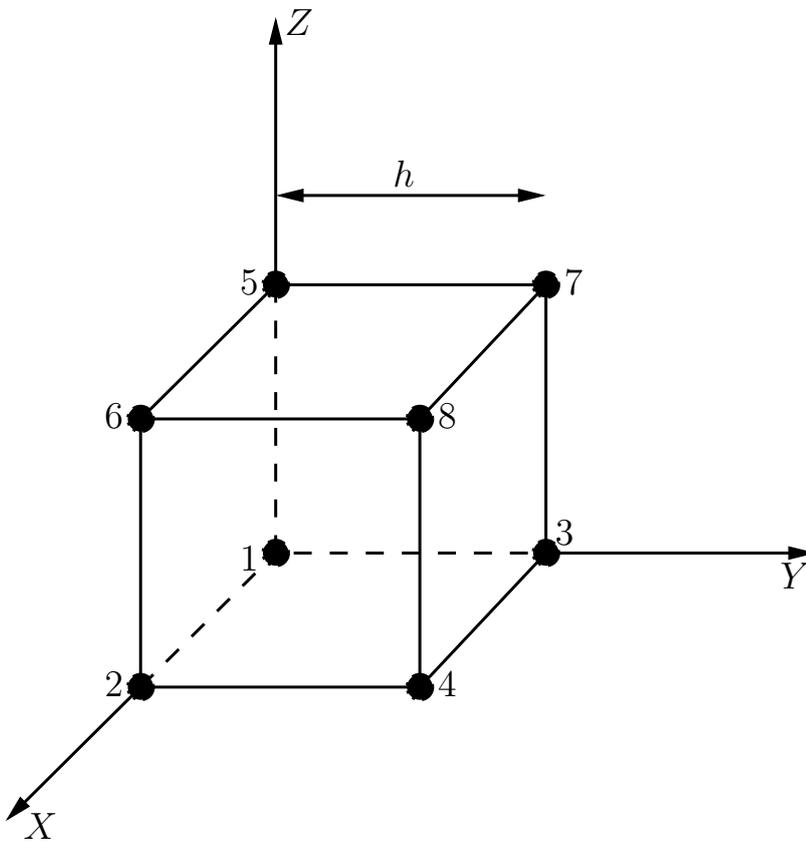


Figure 5.7: Embedding the world cube into \mathbb{R}^3

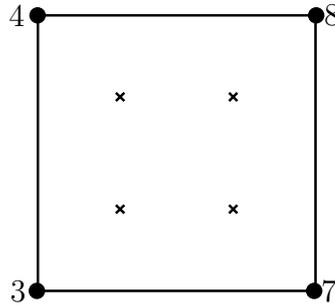


Figure 5.8: The world coordinates of circle centres and corners of the ‘4’ face, mapped to the unit square

The face is mapped into the unit square by the mapping $(X, Y, Z) \mapsto (Z/54, X/54)$. The mapping of the world face, together with the vertex indices is shown in Fig. 5.8

5.2.4.3 Matching image and world features

To identify which two or three faces are present in an image, each mapped image face is first rotated three times about the centre of the square by $\pi/2$ to give four different orientations for the image face. Each of these orientations is then compared with each of the six world faces by nearest neighbour matching to identify which face is present. If there is any rotational symmetry in the face, then multiple matches can be found. Consider Fig. 5.9. Each of the four possible orientations of the ‘4’ face mapped from the image matches the reference ‘4’ face mapped from the cube, shown in Fig. 5.8, due to rotational symmetry of this face and its markers. The problem then becomes one of determining which of the four orientations is the correct one.

Once the identity of the two or three faces is known, the correspondences can be made. For the two face case, the two world vertices that fall on both visible faces can be found. For example, in Fig. 5.1a, the ‘4’ and the ‘6’ face is visible. Referring to the last column of Table 5.2, it can be seen that the two vertices in common to face ‘4’ and face ‘6’ are vertices 7 and 8. Similarly, in Fig. 5.1b, once it is known that faces ‘2’, ‘4’, and ‘6’ are visible, it can be seen that the vertex in common to all three faces is vertex 7.

Consider once again Figures 5.8 and 5.9. In Fig. 5.8, the common vertex, vertex 7, is in the $(1, 0)$ position on the unit square. For the image faces, the common vertex is vertex g . Vertex g is in the $(1, 0)$ position in Fig. 5.9b. Therefore the correct orientation is a single rotation by $\pi/2$, and the correspondences between the image and world coordinates can be found directly. The procedure for the two face case works similarly. In this manner, all image and world correspondences can be found.

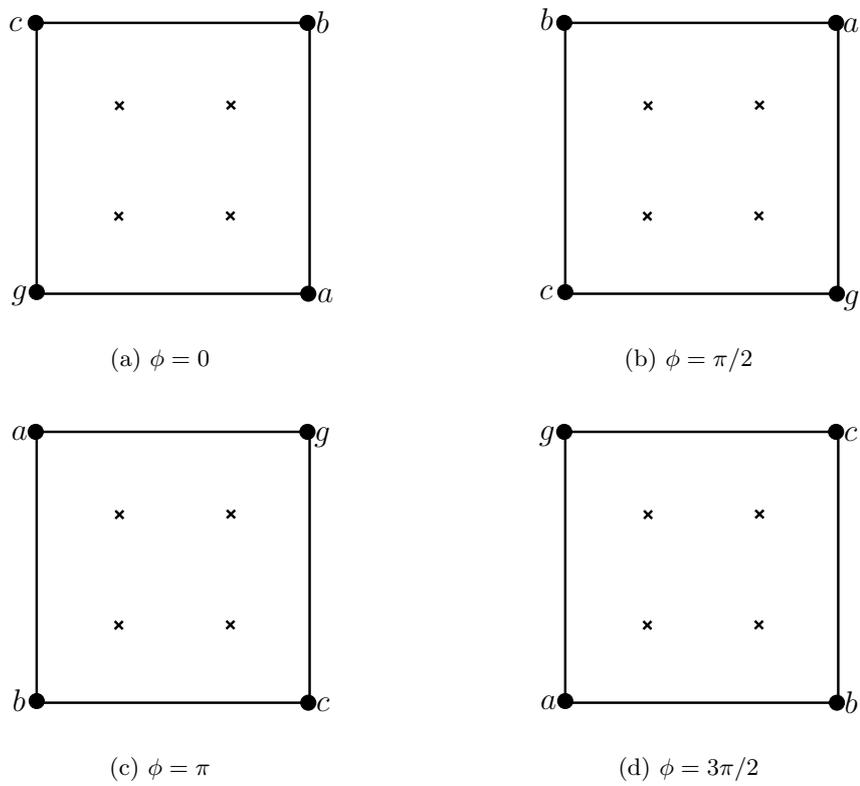


Figure 5.9: The four possible orientations of the ‘4’ face mapped from the image. Each of these four rotations of the ‘4’ face match the world ‘4’ face depicted in Fig. 5.8

5.2.5 Finding image coordinates of circle centres

The centres of the measured ellipses, such as those depicted in Fig. 5.6a, are not unbiased measurements of the projections of the circle centres, as circle centres are not projectively invariant. To obtain an unbiased measurement, the centres of the circles obtained when a face is mapped to the unit square, as described in §5.2.4.1, are computed. These circle centres are then mapped back to the image by H^{-1} , the inverse of the homography used to compute the mapping to the unit square. These mapped points are used as the image measurements of the circle centres.

5.2.6 Camera resection

Once all of the world-image correspondences, $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$, have been made, the camera projection matrix is \mathbf{P} defined to be the matrix which minimises

$$\sum_i d_e(\mathbf{P}\mathbf{X}_i, \mathbf{x}_i)^2 \quad (5.8)$$

where $d_e(\cdot, \cdot)$ is the distance between the Euclidean (\mathbb{R}^2) coordinates of two points. The resulting \mathbf{P} is the maximum likelihood estimate under the assumption of zero world point error, and isotropic Gaussian noise in the image measurements \mathbf{x}_i . Computation of \mathbf{P} is achieved by the standard resection procedure described in 3.5.

5.3 Case study

The cube described in the previous section, while useful for the purposes of illustration, does not have sufficient features to compute a reliable estimate of the camera calibration. A (100mm \times 100mm \times 100mm) cube, shown in Fig. 5.10 was machined from aluminium to an approximate tolerance of 0.1mm. An additional eight circular features were added to each face to improve the accuracy of the resection procedure. The circular features are drilled by a CNC process for accuracy. The cube was chosen to be white to allow the use of a dark background, minimising the effect of shadows. The cube was illuminated by a custom built LED flash mounted on each camera. The images were 2 megapixel images (1600 \times 1200 pixels) taken from standard consumer Canon Powershot G5 cameras.

Each camera was calibrated using the calibration algorithm. Fig. 5.10, panels (b)-(j) show the measured ellipses and constructed face boundaries. Note that the algorithm correctly detected that four out of the five images had three faces present, and constructed the faces accordingly. The outline of the cube and the location and orientation of each camera relative to the cube is depicted in Fig. 5.11.

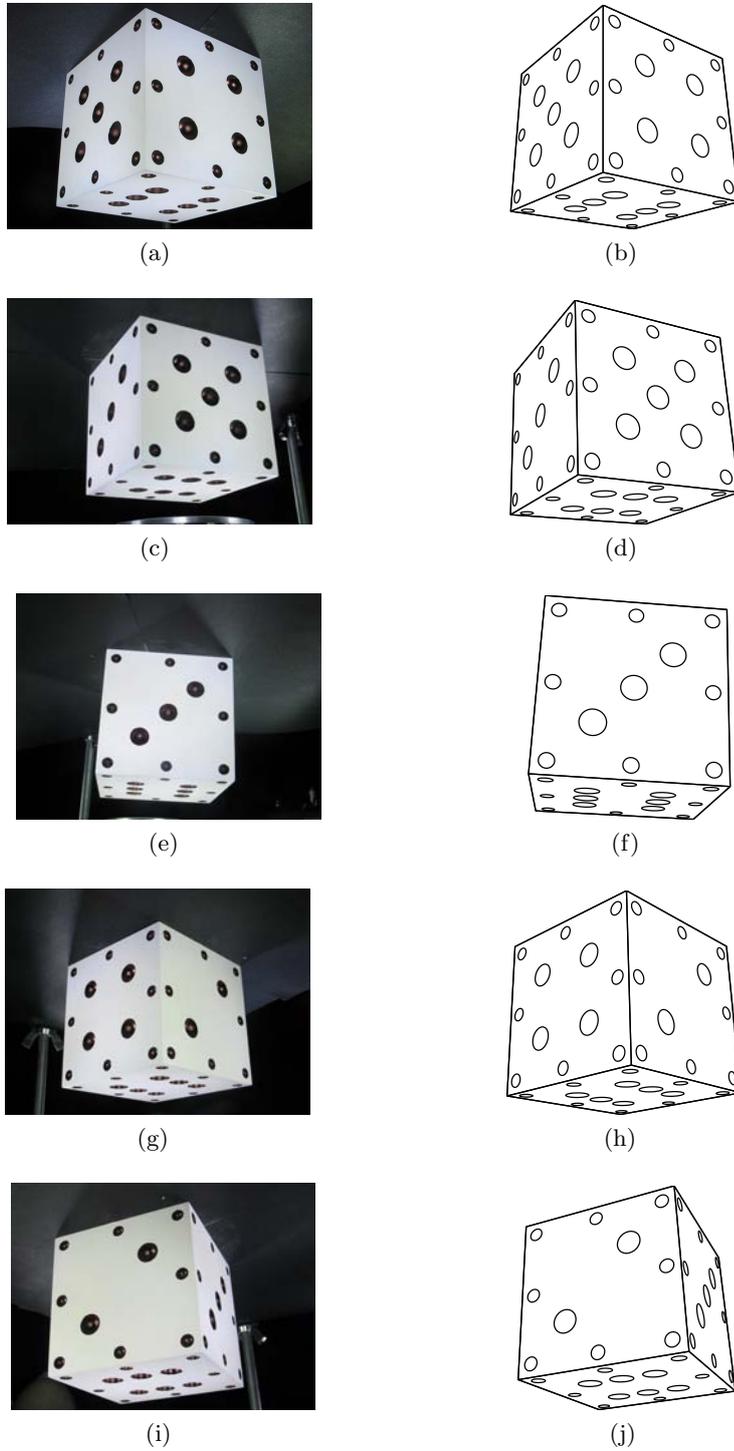


Figure 5.10: Five views of the white calibration cube, together with their fitted elliptical features and face boundaries

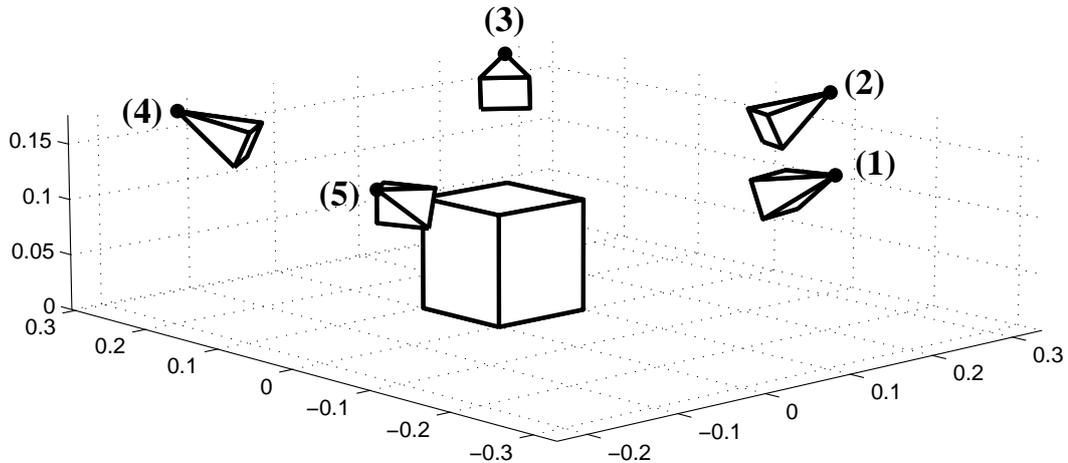


Figure 5.11: Reconstructed camera locations and orientations with respect to the cube

The computed calibration matrix K_1 for the first camera is found to be:

$$K_1 = \begin{bmatrix} 2442 & 0.8941 & 791.4 \\ 0 & 2436 & 590.8 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.9)$$

There are a couple of interesting features in (5.9). The skew, $s = 0.8941$ is not zero. This is not because the pixels are not rectangular. Instead it occurs because there is a small amount of radial distortion present in the image, which is not taken into account in the camera model. The principal point location is about 10 pixels away from the expected $(800, 600)^T$ position in each coordinate. This result shows that the centre of the image is not quite aligned with the optical axis of the camera.

The optical centres of each camera, as can be seen in Fig. 5.11 are shown in Table 5.3, together with the intrinsic calibration matrix K . The cameras are all the same model, using the same settings. This can be seen in the matrices K_i , which all have very similar parameters.

The algorithm was implemented in Matlab and executed on a Pentium M 1.6 GHz laptop computer. The execution time for each camera was around 6 seconds, with a little over 4 seconds being taken up for the image processing stage (§5.2.2). These results suggest that an optimised implementation in a compiled language should run in well under 1.0 seconds.

An experiment was performed to measure the accuracy of the calibration. Each pair of adjacent cameras has a number of image features in common. The image locations of these points, as computed in the calibration procedure in §5.2.4.1, were used to reconstruct the 3D location of the circle centre by triangulation [16]. The difference between the reconstructed points and the true known world points was then computed.

Camera no.	Position (mm)	Calibration matrix K_i
(1)	$\begin{pmatrix} 79.2 \\ -328.1 \\ 173.8 \end{pmatrix}$	$\begin{bmatrix} 2457 & 1.640 & 786.1 \\ 0 & 2453 & 602.0 \\ 0 & 0 & 1 \end{bmatrix}$
(2)	$\begin{pmatrix} 338.1 \\ -69.0 \\ 176.1 \end{pmatrix}$	$\begin{bmatrix} 2467 & 0.945 & 774.0 \\ 0 & 2462 & 594.2 \\ 0 & 0 & 1 \end{bmatrix}$
(3)	$\begin{pmatrix} 282.2 \\ 305.3 \\ 169.2 \end{pmatrix}$	$\begin{bmatrix} 2447 & 1.588 & 792.0 \\ 0 & 2446 & 598.1 \\ 0 & 0 & 1 \end{bmatrix}$
(4)	$\begin{pmatrix} -129.8 \\ 281.2 \\ 169.8 \end{pmatrix}$	$\begin{bmatrix} 2441 & 0.894 & 791.4 \\ 0 & 2436 & 590.8 \\ 0 & 0 & 1 \end{bmatrix}$
(5)	$\begin{pmatrix} -232.6 \\ -110.9 \\ 171.9 \end{pmatrix}$	$\begin{bmatrix} 2433 & -1.177 & 799.7 \\ 0 & 2430 & 609.9 \\ 0 & 0 & 1 \end{bmatrix}$

Table 5.3: Reconstructed 3D coordinates of the optical centre and the intrinsic calibration matrix for each camera

Camera 1	Camera 2	No. of pts reconstructed	Mean error (mm)
1	2	27	0.081
2	3	25	0.086
3	5	25	0.093
5	4	24	0.083
4	1	26	0.076

Table 5.4: Reconstruction results for pairs of adjacent cameras

The results are presented in Table 5.4. In each case, the mean error in reconstruction was less than 0.1mm, near the manufacturing tolerance for the calibration cube. This level of accuracy is more than sufficient for the DIET project [31, 32].

5.4 Discussion and Conclusion

The main contribution of this calibration method is a means of automatically identifying image-world feature correspondences from the a single image of the calibration cube, and thus enabling automatic calibration of the camera. A useful side effect is the ability to easily eliminate projective distortion on each face, allowing accurate unbiased measurements of the centres of circular features on the cube. The projective distortion is traditionally the major drawback to using circular features, usually being corrected

afterwards in an iterative procedure during the resection process [17], however the algorithm developed in this chapter allows for projective distortion correction at the time of measurement.

A specific cube feature pattern has been deliberately left unspecified. Any pattern will work if the configuration of points is unique on each face and there are sufficient points on each face. Avoiding rotational symmetry in choosing a pattern will minimise effort, as well.

The camera model used was the simple linear pinhole projective model, which does not allow for any radial distortion. The standard approaches for dealing with radial distortion, such as choosing radial parameters that make image lines straight are still applicable with a suitably designed configuration of circles on each face. The number of features required to accomplish this task satisfactorily is substantially more than the number on the cube presented in §5.3, common implementations have 100 or more points [17]. Because of the ability to remove projective distortion directly, and thus obtain very accurate image point measurements from circular features, this method, with a more sophisticated camera model, is suitable for high-accuracy applications.

The algorithm presented in this chapter requires only one image, and is fully automatic. Therefore, it overcomes the major drawback of standard chequerboard-based approaches [39, 42], that produce very accurate results, but require numerous images and often a significant amount of user input.

The calibration method presented in this chapter also overcomes all of the shortcomings of the previous method, while retaining the same, or a slightly higher, level of accuracy. Notably, the same reconstruction accuracy levels of under 0.1mm are retained, using the same cameras, despite running the cameras at 2 megapixels rather than 5 megapixels. This accuracy at lower resolution is due to the usage of circular features rather than lines, which can be detected to a high degree of precision. Computational time is reduced from around 2 minutes per camera to around 4 seconds per camera. The need for user input has been eliminated, and the algorithm allows placement of the cameras at a significantly less restrictive range of viewpoints. This calibration algorithm therefore satisfies all of the desiderata set out in §4.1.2.

Chapter 6

FEATURE TRACKING

6.1 Introduction

Feature tracking is an intrinsically local problem, with the objective being to follow the motion of individual points between frames from one camera. The conventional approach is to use an interest point detector, such as the Harris corner detector [15], to identify high-contrast features like corners and areas of high gradient. Once these features are known, they are tracked from image to image by considering a small region around the feature. The location of the feature in the following frame is performed by computing the normalised cross-correlation (NCC) in the window, or by using a gradient descent search.

There are two main disadvantages of methods like these. The first is computational cost, with a large number of correlations needing to be performed. The second, more significant, drawback is that they only consider a small window around the point. If the motion is large enough that other similar points enter the window, these methods will either compute mismatches, or fail to find the match. This second issue is significant in any application with high density points, such as DIET.

Because the features used on the breast surface are artificial fiducials whose image locations can be accurately determined by thresholding, an alternative method for tracking points is to use geometric properties of the point positions, rather than the image intensity data. The simplest case is where the motion is sufficiently small that points can simply be matched by nearest neighbour matching, such as shown in the example in Fig. 6.1. Nearest neighbour matching is akin to the window-based tracking algorithms mentioned previously, as it requires that the matching point be close. This approach is described briefly in §6.2 in the context of the DIET application.

If the motion of points on the surface is sufficiently large that nearest neighbour matching fails, such as shown by example in Fig. 6.2, there are two basic options. First, the frame acquisition rate could be increased, to decrease the amount of motion

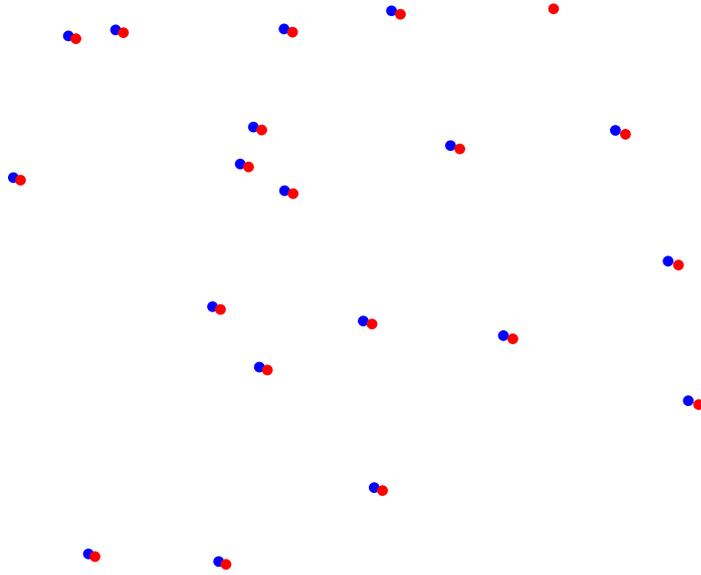


Figure 6.1: Point motion where the motion is significantly less than the point spacing

between frames, but increasing the complexity in hardware and adding computational cost. Alternatively, a more sophisticated approach can be used based on geometric invariants.

If the motion of a small region on the breast surface is considered, the transformation is approximately rigid. This conditions can also be observed at the image level, where points in a small region of the image don't move at random, the neighbourhood structure is preserved. The motion in the image space can therefore be approximated locally by Euclidean or similarity transformations, where the region is allowed to rotate or translate, or scale for similarity transformations. Given a set of such points, it is possible to compute a normalising transformation, similar to the moving frame of differential geometry [10, 11], that provides a Euclidean (similarity) invariant representation of the set of points, comprising joint invariants of the set of points. This invariant representation is known as its (discrete) *signature*, similar in spirit to the signatures defined by Calabi *et. al* [2] and Olver [29]. Points can then be matched in this signature frame, as the transformation has effectively been “undone”, creating a potentially easier matching problem.

To obtain an ordered set of points, points of three colours are used: red, green, and blue. An ordered set of three points is defined to be a red point and its nearest green and blue neighbours. This approach can be extended to more colours to get larger point sets, but three is particularly convenient because of how easily points in these colours are segmented from colour images.

The signature is defined for the red points only. A similar point triple definition can be used to match green and blue points. However, these correspondences can easily

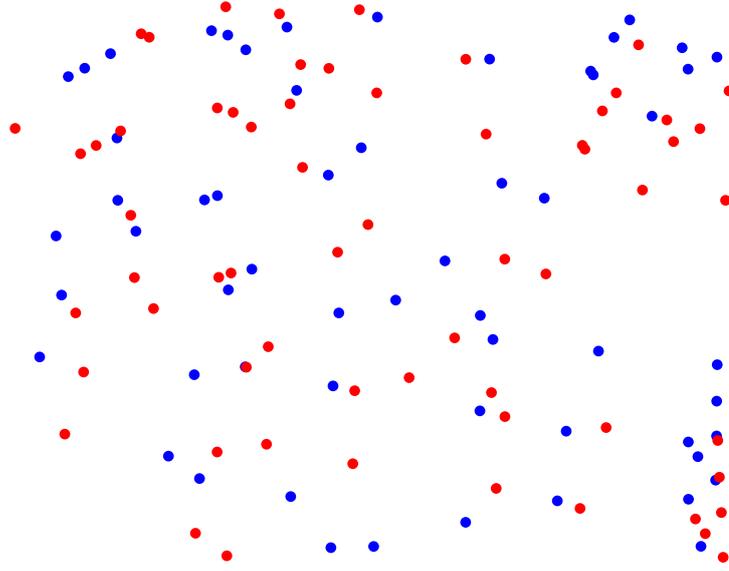


Figure 6.2: Point motion where the motion is larger than the point spacing

be found once the red point matches are known. The transformation of the red points can be interpolated onto the remaining green and blue points, allowing the remainder of the points to be matched directly by nearest neighbour matching.

6.2 Nearest neighbour matching: small motion

If the motion of the points is significantly less than the point spacing, such as in the example shown in Fig. 6.1, then the algorithm for feature tracking is trivial. Features can then be matched by nearest neighbour matching. The algorithm, included for completeness, shown in Algorithm 6.1. Note that this algorithm requires a symmetric nearest neighbour relationship.

Algorithm 6.1 Nearest neighbour matching

Let two feature sets be \mathcal{S} , \mathcal{S}' . For a given known upper bound δ on the magnitude of the transformation:

1. For each point $\mathbf{x}_i \in \mathcal{S}$, find the closest point $\mathbf{x}'_j \in \mathcal{S}'$ by Euclidean distance, subject to $\|\mathbf{x}'_j - \mathbf{x}_i\| < \delta$, if such a point exists.
 2. For each point $\mathbf{x}'_i \in \mathcal{S}'$, find the closest point $\mathbf{x}_j \in \mathcal{S}$ by Euclidean distance, subject to $\|\mathbf{x}_j - \mathbf{x}'_i\| < \delta$, if such a point exists.
 3. If the closest point obtained in this manner to \mathbf{x}_i is \mathbf{x}'_j and the closest point to \mathbf{x}'_j is \mathbf{x}_i , then the points \mathbf{x}_i and \mathbf{x}'_j are said to correspond.
-

6.3 Invariant signatures

6.3.1 Euclidean-invariant signature for ordered point sets

A Euclidean transformation is a composition of a rotation and a translation. In two dimensions, it can be parametrised by three parameters: the rotation angle θ , and the two components of the translation vector $\mathbf{t} = (t_x, t_y)^\top$. A Euclidean transformation can be written in matrix form as:

$$\mathbf{x}' = \mathbf{R}(\theta)\mathbf{x} + \mathbf{t} \quad (6.1)$$

where $\mathbf{R}(\theta)$ is a rotation matrix:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (6.2)$$

Under Euclidean transformations, distances and angles are preserved. Given an ordered set of points, $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, they can be transformed by a Euclidean transformation:

$$\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i + \mathbf{t}. \quad (6.3)$$

There is a systematic means of computing a set of invariants for the set of points by means of a moving frame. The set of invariants constitutes a *signature* for the set of points $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, and it will be shown that the signature is the same for any two sets of points differing only by a Euclidean transformation. Hence, they can be used to match ordered sets of points under Euclidean transformations.

The signature is constructed in two steps:

1. Translate the points so that the first point in the translated coordinates, $\mathbf{x}_1^{(t)}$, is the origin:

$$\mathbf{x}_i^{(t)} = \mathbf{x}_i - \mathbf{x}_1 \quad (6.4)$$

2. Rotate the translated points about the origin so that the second point in the new transformed coordinates, $\mathbf{x}_2^{(e)}$ lies on the positive x -axis:

$$\mathbf{x}_i^{(e)} = \mathbf{R}(-\theta)\mathbf{x}_i^{(t)} \quad (6.5)$$

where θ is the counterclockwise angle between the positive x -axis and $\mathbf{x}_2^{(t)}$, and $\mathbf{R}(-\theta)$ is a rotation matrix as defined in (6.2).

The transformation from coordinates \mathbf{x}_i to $\mathbf{x}_i^{(e)}$ is a single Euclidean transformation:

$$\mathbf{x}^{(e)} = \mathbf{R}(-\theta)\mathbf{x} + \mathbf{t} \quad (6.6)$$

where $\mathbf{R}(-\theta)$ is the same as defined previously, and $\mathbf{t} = -\mathbf{R}(-\theta)\mathbf{x}_1$. Note that the parameters of the transformation are only dependent on two points: \mathbf{x}_1 , and \mathbf{x}_2 . This *normalising* transformation is unique if \mathbf{x}_1 and \mathbf{x}_2 are distinct. In the new coordinate system, the transformed coordinates are given by:

$$\mathbf{x}_1^{(e)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{x}_2^{(e)} = \begin{pmatrix} x_2^{(e)} \\ 0 \end{pmatrix} \quad \mathbf{x}_i^{(e)} = \begin{pmatrix} x_i^{(e)} \\ y_i^{(e)} \end{pmatrix} \quad i = 1, \dots, 3. \quad (6.7)$$

Define the following quantities:

$$\begin{aligned} I_i &= x_i^{(e)} & i &= 2, \dots, n \\ J_j &= y_j^{(e)} & j &= 3, \dots, n. \end{aligned} \quad (6.8)$$

These $2n - 3$ quantities are invariants of the set $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ under Euclidean transformations, and define the following signature mapping $s_e: (\mathbb{R}^2)^{\times n} \mapsto \mathbb{R}^{2n-3}$, written:

$$s_e(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (I_2, I_3, J_3, \dots, I_n, J_n)^\top \quad (6.9)$$

The vector $(I_2, I_3, J_3, \dots, I_n, J_n)^\top$ is called the signature of the set $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$. The invariance under Euclidean transformations is stated in the following Theorem.

Theorem 6.1. *Two ordered sets of points $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, $(\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_n)$ are related by a Euclidean transformation if and only if they have the same signature*

Proof. If the signatures are the same, then there exist unique normalising transformations for each set, parametrised by \mathbf{R}, \mathbf{t} and \mathbf{R}', \mathbf{t}' , such that

$$\mathbf{R}\mathbf{x}_i + \mathbf{t} = \mathbf{R}'\mathbf{x}'_i + \mathbf{t}',$$

which can be rearranged

$$\mathbf{x}'_i = \mathbf{R}'^\top \mathbf{R}\mathbf{x}_i + \mathbf{R}'^\top (\mathbf{t} - \mathbf{t}')$$

which is a Euclidean transformation.

Now, let the two sets be related by a Euclidean transformation,

$$\mathbf{x}_i = \mathbf{R}_e \mathbf{x}'_i + \mathbf{t}_e. \quad (6.10)$$

Let the normalising transformation for $\{\mathbf{x}_i\}$ be parametrised by $\mathbf{R}_n, \mathbf{t}_n$. Applying this transformation to both sides of (6.10) gives

$$\mathbf{R}_n \mathbf{x}_i + \mathbf{t}_n = \mathbf{R}_n \mathbf{R}_e \mathbf{x}'_i + (\mathbf{R}_n \mathbf{t}_e + \mathbf{t}_n). \quad (6.11)$$

The LHS is the unique normalisation for $\{\mathbf{x}_i\}$. The RHS is a Euclidean transformation, and because the LHS is a set of normalised coordinates, the RHS must be the unique

set of normalised coordinates for $\{\mathbf{x}'_i\}$. Therefore both sets of points have the same normalised coordinates. \square

6.3.2 Similarity-invariant signature for ordered point sets

Similarity transformations are like Euclidean transformations, except they allow an overall scale factor, and thus depend on four parameters instead of three. Therefore, given a set of n points, with $2n$ degrees of freedom in choosing their coordinates, there should be $2n - 4$ functionally independent invariants under similarity transformations.

A similarity transformation can be represented by a matrix-vector equation:

$$\mathbf{x}' = s\mathbf{R}(\theta)\mathbf{x} + \mathbf{t} \quad (6.12)$$

where s is a constant scale factor, and \mathbf{R}, \mathbf{t} are as defined previously. The procedure for deriving a signature is analogous to the presentation in §6.3.1, with the same translation and rotation components. Starting, then, with the normalising Euclidean transformation of (6.6), defined:

$$\mathbf{x}^{(e)} = \mathbf{R}(-\theta)\mathbf{x} + \mathbf{t}. \quad (6.13)$$

The normalising similarity transformation is obtained by scaling all coordinates by $x_2^{(e)}$, so that the resulting $\mathbf{x}_2^{(s)} = (1, 0)^\top$. The transformation is defined:

$$\mathbf{x}_i^{(s)} = \frac{1}{x_2^{(e)}} \mathbf{x}_i^{(e)} \quad i = 1, \dots, n. \quad (6.14)$$

In this coordinate system, the transformed points are given by:

$$\mathbf{x}_1^{(s)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{x}_2^{(s)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{x}_k^{(s)} = \begin{pmatrix} x_k^{(s)} \\ y_k^{(s)} \end{pmatrix} \quad k = 1, \dots, 3 \quad (6.15)$$

The $2n - 4$ similarity invariants are therefore:

$$\begin{aligned} I_i &= x_i^{(s)} & i &= 3, \dots, n \\ J_j &= y_j^{(s)} & j &= 3, \dots, n \end{aligned} \quad (6.16)$$

These invariants define a signature mapping $s_s: (\mathbb{R}^2)^{\times n} \mapsto \mathbb{R}^{2n-4}$, written:

$$s_s(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (I_3, J_3, \dots, I_n, J_n)^\top \quad (6.17)$$

Therefore, two ordered sets of n points $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ and $(\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_n)$ differ by a similarity transformation if and only if $s_s(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = s_s(\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_n)$. Note that the group of similarity transformations includes the group of Euclidean transformations, and thus the similarity invariant is automatically an Euclidean invariant

as well. It should be noted that, like the Euclidean signature, the normalising transformation for the signature is only dependent on the first two points \mathbf{x}_1 and \mathbf{x}_2 .

6.4 Point tracking with a 3-point Euclidean invariant signature

To make use of the signatures described in §6.3.1, §6.3.2, a means of choosing ordered point sets from among the image points is required. Assume that the image features are of three colours: red, green, and blue. An ordered set of three points can be constructed by choosing a red point and then finding the nearest green and blue points. Denote the set of feature points as:

$$\mathcal{I} = \{\mathbf{x}_i\} \quad i = 1, \dots, n \quad (6.18)$$

Each point \mathbf{x}_i has a colour $c_i \in \{1, 2, 3\}$ where $c_i = 1$ represents red points, 2 represents green points, and 3 represents blue points.

Let $\mathcal{R}, \mathcal{G}, \mathcal{B}$ be the sets of red, green, and blue points, respectively:

$$\begin{aligned} \mathcal{R} &= \{\mathbf{x}_i \in \mathcal{I} \mid c_i = 1\} \\ \mathcal{G} &= \{\mathbf{x}_i \in \mathcal{I} \mid c_i = 2\} \\ \mathcal{B} &= \{\mathbf{x}_i \in \mathcal{I} \mid c_i = 3\} \end{aligned} \quad (6.19)$$

A mapping $f: \mathcal{R} \mapsto \mathcal{R} \times \mathcal{G} \times \mathcal{B}$ that maps \mathcal{R} onto ordered triples of points is then defined as follows:

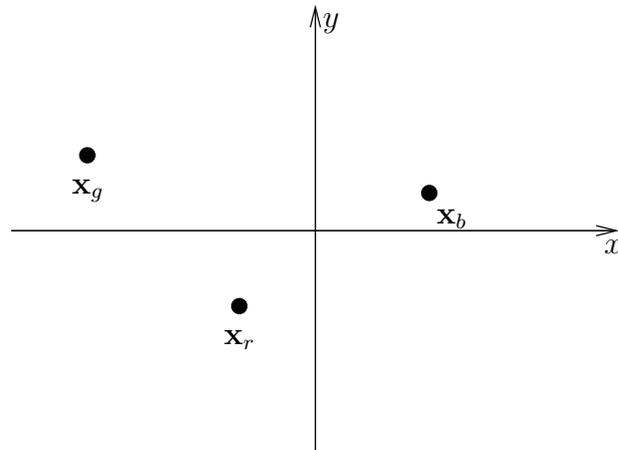
$$f(\mathbf{x}_r) = (\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b) \quad (6.20)$$

where \mathbf{x}_g is the nearest green point to \mathbf{x}_r , and \mathbf{x}_b is the nearest blue point to \mathbf{x}_r . Therefore, a Euclidean invariant signature $s_r: \mathcal{R} \mapsto (\mathbb{R}^3)^{\times k}$, where k is the number of red points, is constructed as follows:

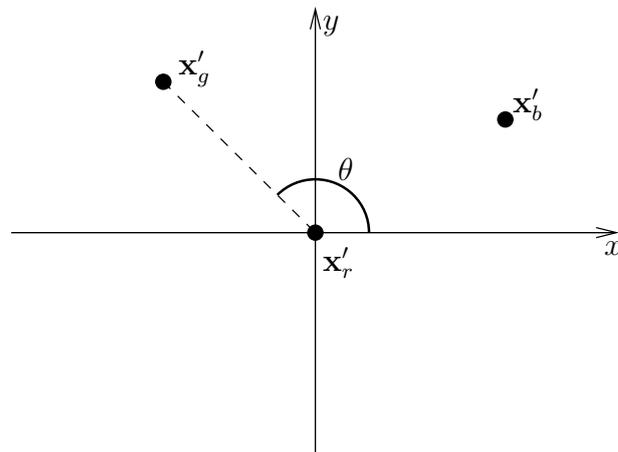
$$s_r(\mathbf{x}_r) = s_e(f(\mathbf{x}_r)) \quad (6.21)$$

where s_e is the signature mapping defined in (6.9). Because nearest neighbours are invariant under Euclidean transformations, this signature is also invariant under Euclidean transformations. Construction of the signature given the ordered triple $(\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b)$ is shown in Fig. 6.3.

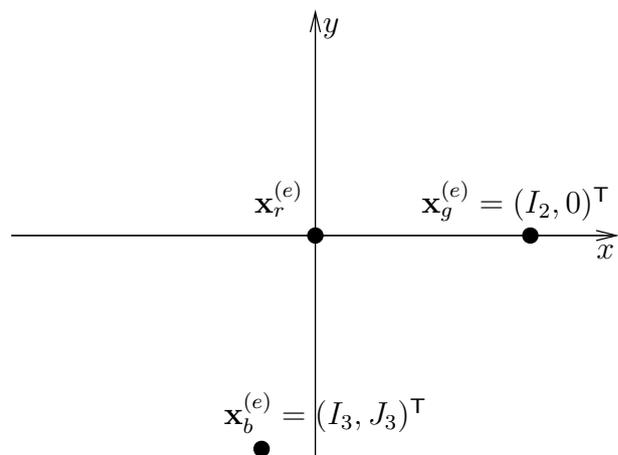
This signature has only three components. In the presence of noise, it is inevitable that if signatures are computed for hundreds or thousands of points that the signature space will become overcrowded, and that mismatches will occur. However, when performing feature tracking, it is almost always possible to estimate an upper bound r_m on the magnitude of the transformation between frames, and this bound can be used



(a) Ordered triple $(\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b)$ consisting of \mathbf{x}_r and its nearest green and blue neighbours



(b) Translated points, so \mathbf{x}'_r is the origin



(c) Points rotated by θ so that $\mathbf{x}_g^{(e)}$ is on the x -axis, yielding the invariant $(I_2, I_3, J_3)^\top$

Figure 6.3: Constructing the 3-point Euclidean invariant

to constrain the signature matching process. Another potential problem is that in the presence of noise, nearest neighbours may not be preserved if the two nearest neighbours of a given colour are a similar distance from the point. Therefore, signatures are only computed for points if the nearer of the two nearest neighbours of each colour is at least a specified threshold distance d nearer than the farther.

The basic idea of the tracking procedure is that suitable red points are matched by their invariant signatures. The remaining unmatched points from the first frame are then transformed into the second frame by interpolating the motion of the red points. The transformed points can then be matched by nearest neighbour matching. The final resulting algorithm for tracking features using the 3-point Euclidean invariant signature is presented in Algorithm 6.2

6.4.1 3-point ordered similarity invariant

A 3-point similarity invariant signature can be constructed analogously to the Euclidean-invariant signature constructed in §6.4, with the signature mapping, s_r (6.21) being replaced with:

$$s_r(\mathbf{x}_r) = s_s(f(\mathbf{x}_r)) \quad (6.22)$$

where s_s is the mapping of (6.17). Because finding nearest neighbours is similarity invariant, f , and thus the signature mapping (6.22), is also similarity invariant. The matching algorithm can similarly be defined the same way as Algorithm 6.2 by changing the signature definition.

The reason why it might be useful to use a similarity invariant, rather than a Euclidean invariant, is because the similarity invariant allows changes of scale in the images. Changes of scale occur when the object being imaged moves toward or away from the camera. Therefore, if a significant amount of the motion is parallel to the optical axis of the camera, the similarity signature may give better results.

Because of the additional degree of freedom present in similarity transformations, the signature only has two components, rather than the three components of the Euclidean signature. The signature space is therefore smaller, and potentially more likely to generate mismatches due to the space being filled. If an additional colour was used, for example black points, invariants using four points could be constructed, which would have four components, solving this problem.

6.5 A similarity-invariant approach without ordered points

The normalising transformation (6.14) used to construct similarity invariants is itself constructed from only two points, \mathbf{x}_1 and \mathbf{x}_2 , with the coordinates of the transformed

Algorithm 6.2 Feature tracking using a 3-point Euclidean colour-based invariant

Input:

- Two sets of features $\mathcal{I}, \mathcal{I}'$ to be matched, consisting of red, green, and blue points

Parameters:

- An upper bound r_m on the magnitude of the transformation of a point between $\mathcal{I}, \mathcal{I}'$
- The minimum difference d between the distances to the nearest and second nearest green neighbours, and also the nearest and second nearest blue neighbours required to compute the signature for a red point

Algorithm:

1. Form the sets of coloured points $\mathcal{R}, \mathcal{G}, \mathcal{B}$ from \mathcal{I} , and $\mathcal{R}', \mathcal{G}', \mathcal{B}'$ from \mathcal{I}' as described in (6.19), and define the signature mappings s_r and s'_r accordingly, as in (6.9)
 2. For each red point $\mathbf{x} \in \mathcal{R}$ whose nearest green neighbour is at least distance d closer to \mathbf{x} than the next nearest green neighbour:
 - (a) Find the set of red points in \mathcal{R}' within r_m of \mathbf{x} by Euclidean distance
 - (b) From among these points, find the point $\mathbf{x}' \in \mathcal{R}'$ whose signature $s'_r(\mathbf{x}')$ is closest to $s_r(\mathbf{x})$ by Euclidean distance, provided the distance is less than some threshold t
 - (c) If such a match is found, points \mathbf{x} and \mathbf{x}' are said to match, and the vector mapping \mathbf{x} to \mathbf{x}' (by vector addition) is $\mathbf{u} = \mathbf{x}' - \mathbf{x}$
 3. The set of points $\mathbf{x} \in \mathcal{R}$ that are successfully matched, together with their associated transformation vectors \mathbf{u} , form a discrete vector field.
 4. The estimated locations $\bar{\mathbf{x}}'$ in the second image of the unmatched points in the first image can be found by interpolating the discrete vector field formed from the red point matches and translating the points accordingly.
 5. The points $\bar{\mathbf{x}}'$ can be matched with the points \mathbf{x}' by nearest neighbour matching, as in Algorithm 6.1
-

remaining points being the components of the similarity invariant signature (6.17). Given two unordered sets of points $\{\mathbf{x}_i\}$, $\{\mathbf{x}'_i\}$, known to be related by a similarity transformation, and a method of choosing two matching points from each set, the normalising transformation (6.14) parametrised by s , \mathbf{R} and \mathbf{t} is defined:

$$\mathbf{x}^{(s)} = s\mathbf{R}\mathbf{x} + \mathbf{t}. \quad (6.23)$$

This transformation can be computed from the two points for each set, and will map matching points in $\{\mathbf{x}_i\}$ and $\{\mathbf{x}'_i\}$ to the same normalised coordinates, essentially undoing the similarity transformation.

This idea can be used to develop a point-matching procedure. Define $\mathcal{I}, \mathcal{I}'$ to be the two sets of points to be matched. Assume that, locally, the points of \mathcal{I} and \mathcal{I}' are related by a similarity transformation. For a point $\mathbf{x} \in \mathcal{I}$, a similarity normalising transformation (6.14) can be computed from \mathbf{x} and its nearest neighbour $\mathbf{y} \in \mathcal{I}$. This transformation can then be used to map a neighbourhood of \mathbf{x} into normalised coordinates.

To find the matching point to \mathbf{x} in \mathcal{I}' , the neighbourhood of each point $\mathbf{x}' \in \mathcal{I}'$ is mapped to normalised coordinates in the same way. If a point $\mathbf{x}' \in \mathcal{I}'$ is the matching point to \mathbf{x} , then their respective nearest neighbours \mathbf{y}', \mathbf{y} will also be matching points, by the invariance of nearest neighbours under similarity transformations. Therefore, the two constructed normalising transformations will mean that a point \mathbf{u} in the neighbourhood of \mathbf{x} will be mapped to the same coordinates as the corresponding point \mathbf{u}' in the neighbourhood of \mathbf{x}' . As a result, if \mathbf{x} matches \mathbf{x}' , their normalised neighbourhoods will consist of overlapping points.

Because of the changes of scale allowed in a similarity transformation, the neighbourhood of a point has to be defined in a manner that allows for changes of scale. Therefore, rather than using the usual disk of radius r about \mathbf{x} to define the neighbourhood, a disk of radius $k\|\mathbf{x} - \mathbf{y}\|$ is used, where \mathbf{y} is the nearest neighbour to \mathbf{x} and k is a scalar.

In the presence of noise, if the distance to the nearest neighbour \mathbf{y} to \mathbf{x} is almost the same as the distance to the next nearest neighbour \mathbf{v} , then it is possible that the two may be accidentally transposed. If two different points are used to construct the normalised neighbourhoods, the neighbourhoods will not overlap and the matching will fail. To ensure this problem does not occur, matching is only attempted on points whose nearest neighbour is significantly closer than their next nearest neighbour.

The resulting matching algorithm is presented in Algorithm 6.3.

Algorithm 6.3 Pattern matching using similarity invariant neighbourhoods

Input:

- Two point sets $\mathcal{I}, \mathcal{I}'$

Parameters:

- k : scale factor used to determine neighbourhood radius
- d_{key} : distance threshold required for a feature to be selected as a keypoint
- t : pattern matching threshold
- m : minimum number of neighbourhood point matches required for a positive match

Algorithm:

1. For each point $\mathbf{x} \in \mathcal{I}$
 - (a) Compute the two nearest neighbours \mathbf{y}, \mathbf{v} to \mathbf{x}
 - (b) if $|\|\mathbf{y} - \mathbf{x}\| - \|\mathbf{v} - \mathbf{x}\|| > d_{\text{key}}$, then select \mathbf{x} as a keypoint, and store the nearest neighbour \mathbf{y}
 2. For each keypoint \mathbf{x} with nearest neighbour \mathbf{y}
 - (a) Construct the normalising transformation (6.14) from \mathbf{x} and \mathbf{y}
 - (b) Find all points within a radius $k\|\mathbf{x} - \mathbf{y}\|$ of \mathbf{x} , and transform them by the normalising equation, and store the resulting coordinates
 3. Repeat steps 1 and 2 for points $\mathbf{x}' \in \mathcal{I}'$.
 4. For a keypoint $\mathbf{x} \in \mathcal{I}$
 - (a) Compare the stored normalised neighbourhood with each neighbourhood of the keypoints of \mathcal{I}' by nearest neighbourhood matching with a distance threshold t .
 - (b) The keypoint \mathbf{x}' with the most point matches within the distance threshold t is the matching point to \mathbf{x} . If maximum number of matches obtained in this way is less than m , then no match for \mathbf{x} has been found.
 5. A motion field can be constructed and interpolated in the same way as in Algorithm 6.2 to match the remaining unmatched points by nearest neighbour matching
-

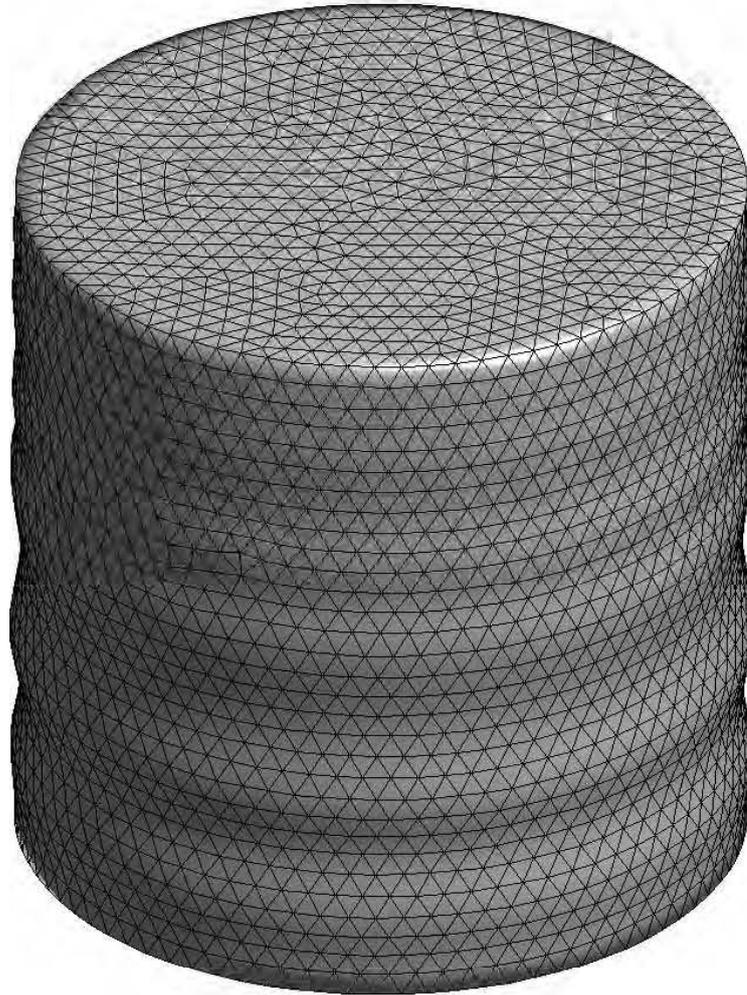


Figure 6.4: Finite element gel cylinder model used for simulation, showing one frame of the cylinder in steady state motion

6.6 Proof of concept experiments

6.6.1 Computer simulation

6.6.1.1 *Motion model*

To simulate the motion of the feature points, a finite element model of a gel cylinder with similar elastic properties to human breast tissue was used [31]. The 17000 nodal finite element mesh was developed in GAMBIT, and the model simulated in Fortran 90, the matrix inversion being done with a direct sparse matrix inversion package, MUMPS. The model was simulated with a 50Hz actuation frequency, with amplitude 0.5mm. Fig. 6.4 shows a picture of the cylinder steady state deformation and meshing.

Coloured points in approximately equal ratio were randomly applied to the surface

by taking samples of a Poisson process of density $\rho = 20\text{pts}/\text{cm}^2$ on the cylinder surface. Their motion was computed by interpolating the motion of the surface nodes of the cylinder mesh onto the random points using a scattered data surface fitting routine, `gridfit` [8], which is based on a modified ridge estimator. To simulate the image capture system, 20 frames were generated, equally spaced by phase offset of 18° over one cycle.

The image coordinates were determined by using a pinhole camera model, using a camera situated at $(30, 0, 20)^\top$ cm, where the cylinder has radius 3.4cm, height 7.4cm, and the base centred on the origin and aligned with the positive z -axis. The camera was defined to have an *a priori* calibration matrix:

$$\mathbf{K} = \begin{bmatrix} 3000 & 0 & 800 \\ 0 & 3000 & 600 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.24)$$

These parameter values for \mathbf{K} use parameters similar to cameras calibrated in actual laboratory experiments, such as discussed in Chapter 5. Gaussian noise of varying standard deviations was added to the image measurements.

A simulated image of the cylinder from the camera is depicted in Fig. 6.5 and image trajectories of some individual points with the different levels of noise used are shown in Fig. 6.6. The trajectories of all the points for the noise-free case are shown in Fig. 6.7.

6.6.1.2 Validation of Euclidean assumption

To test the validity of the assumption that the transformation between frames is locally Euclidean, and hence the validity of using the Euclidean-invariant signature, small groups of 5-6 neighbouring points were randomly selected and the motion between different pairs of frames was examined. A nonlinear least squares estimate of the best Euclidean transformation between the two groups, parameterised by a rotation angle, θ , and two translation components, t_x, t_y , was performed. The cost function for this minimisation was defined:

$$c(\theta, t_x, t_y) = \sum_i \|R(\theta)\mathbf{x}_i + \mathbf{t} - \hat{\mathbf{x}}_i\|^2 \quad (6.25)$$

where $R(\theta)$ is the standard rotation matrix parametrised by the counterclockwise rotation angle θ , the vectors \mathbf{x}_i are points in the first image with $\hat{\mathbf{x}}_i$ the corresponding points in the second image, and \mathbf{t} is the translation vector $[t_x, t_y]^\top$. The points \mathbf{x}_i were transformed by the estimated Euclidean transformation, and the distance to each corresponding point, $\hat{\mathbf{x}}_i$ computed.

Typical results, for any combination of 2 frames from the 20, were a mean distance

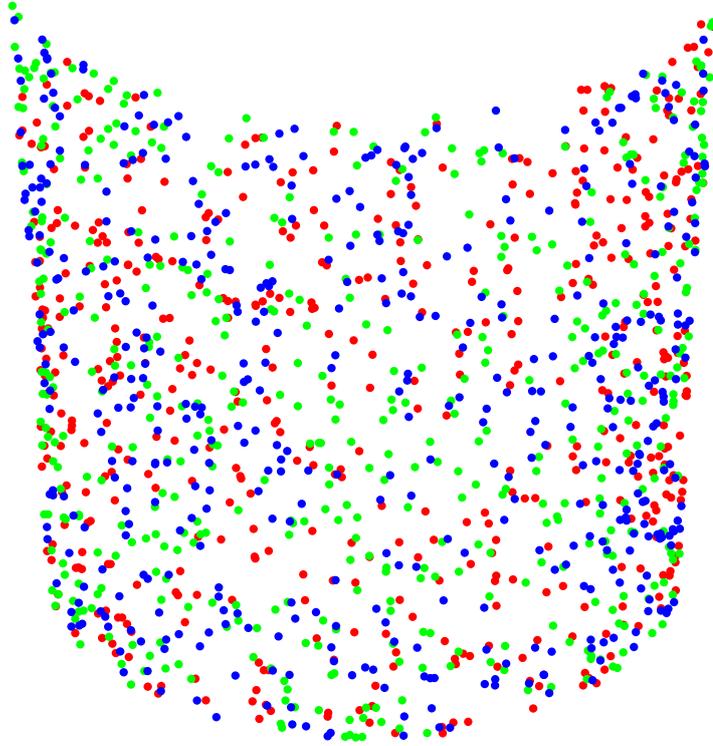


Figure 6.5: Points on the cylinder as measured from the camera

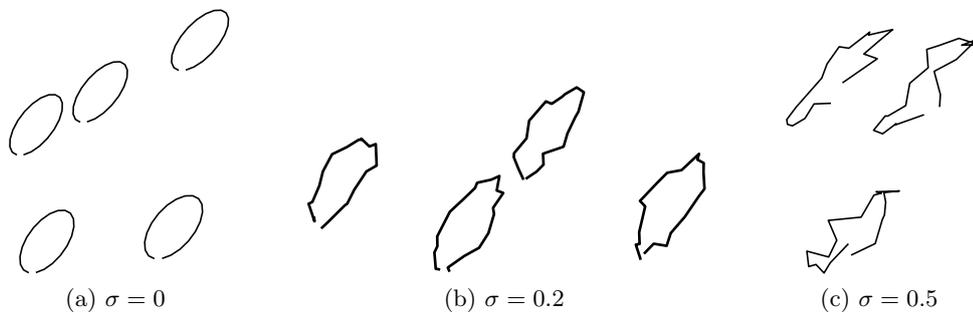


Figure 6.6: Image point trajectories for different levels of measurement noise. The noise is Gaussian with standard deviation σ in each coordinate

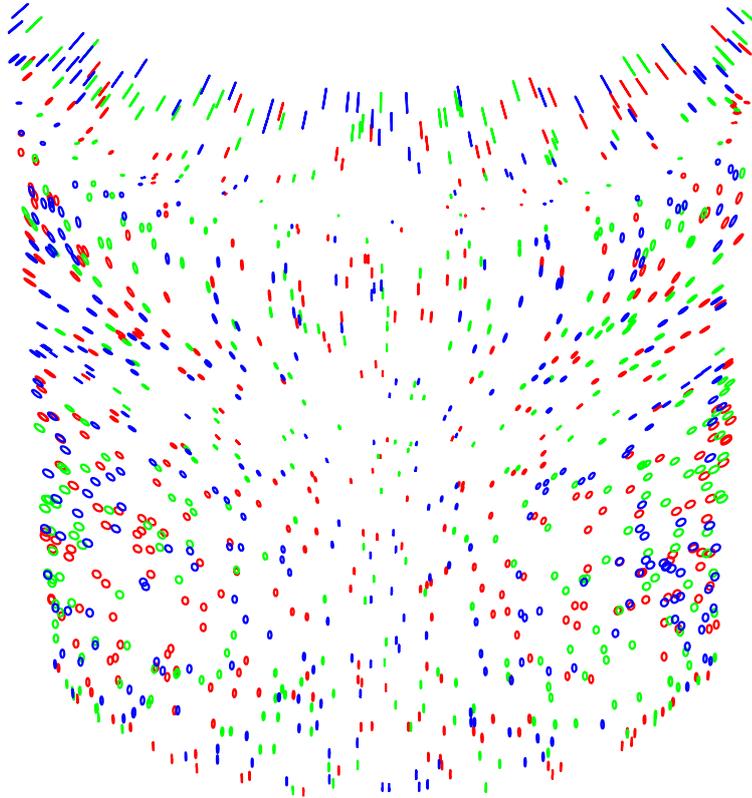


Figure 6.7: The motion of each of the feature points

error of less than 0.1 pixels, where the transformations ranged in magnitude from 1 to 15 pixels. Therefore, the transformations caused by the motion and deformation of the simulated surface can be well-approximated locally by Euclidean transformations.

6.6.1.3 Experiments

Three tracking algorithms were tested:

- 3-point colour-based Euclidean invariant signature tracking, which uses Algorithm 6.2 with the Euclidean invariant. The following parameters were used: $r_m = 20$ pixels, $d = 4$ pixels, a signature matching threshold of 2 pixels, and nearest-neighbour matching threshold of 5 pixels for the interpolated points.
- 3-point colour-based similarity invariant signature tracking, which uses Algorithm 6.2 with the similarity invariant. The parameters used were the same as for the Euclidean invariant, except that the signature matching threshold is 0.5
- Similarity invariant neighbourhood matching, which uses Algorithm 6.3. The parameters used were: $d_{\text{key}} = 5$ pixels, $t = 1$, and $m = 5$. Instead of using the variable radius, a constant neighbourhood radius of 30 pixels was used

	Euclidean signature	Similarity signature	Similarity neighbourhood
Matching:			
n	1160	1160	1160
n_m	159	169	206
n_b	1	1	0
p_c	99.7	99.2	97.3
p_0	99.7	99.6	99.7
Tracking:			
n	1160	1160	1160
n_t	1160	1154	1160

Table 6.1: Results for $\sigma = 0$

The algorithms were tested at the three different noise levels of $\sigma = 0$, $\sigma = 0.2$, and $\sigma = 0.5$ pixels in each coordinate. Two experiments were performed for each algorithm and each noise level:

1. Points were matched from two frames with phase difference 180° , thereby testing the algorithms on a relatively large-magnitude transformation. The following quantities were recorded:
 - The number n of feature points present
 - The number n_m of signature points corresponded
 - The number n_b of mismatched signature points
 - The percentage p_c of points for which a correspondence is made, including by interpolation
 - The percentage p_0 of the correspondences which are correctly matched
2. Features were tracking from all adjacent frames, from the first frame back around to the first again. The following quantities are recorded:
 - The number n of feature points present
 - The number n_t of feature points tracked all the way around one cycle

6.6.1.4 Results

The results for each experiment are listed in Tables 6.1, 6.2, and 6.3.

From the tables, it is clear that the performance of the three algorithms is approximately equivalent. They also degrade in a similar manner under noise. It is interesting to note that the 3-point similarity invariant performs slightly worse in all tests than the 3-point Euclidean invariant.

This performance drop occurs for two reasons. First, the process of dividing by the scaling factor introduces an overall scale error dependent on the relative error in

	Euclidean signature	Similarity signature	Similarity neighbourhood
Matching:			
n	1160	1160	1160
n_m	161	157	147
n_b	1	2	0
p_c	99.0	94.5	97.2
p_0	99.3	99.3	99.7
Tracking:			
n	1160	1160	1160
n_t	1121	987	1114

Table 6.2: Results for $\sigma = 0.2$

	Euclidean signature	Similarity signature	Similarity neighbourhood
Matching:			
n	1160	1160	1160
n_m	128	133	85
n_b	4	6	1
p_c	95.1	93.9	89.8
p_0	99.2	98.9	98.9
Tracking:			
n	1160	1160	1160
n_t	616	386	510

Table 6.3: Results for $\sigma = 0.5$

the scaling factor. This scaling error is in addition to the already existing measurement noise, and thus degrades the noise performance. The second reason is due to the dimensionality of the signature space. The advantage of having a wider class of transformations available is offset by the disadvantage of having a signature space that is one dimension smaller, and noise in this space is thus more likely to induce mismatches than in the three-dimensional Euclidean signature space.

The colour-free tracking algorithm that relies on the similarity invariant neighbourhood performs approximately as well as the Euclidean colour-based invariant signature. However, in the Matlab implementation used, it is about 1.5 times slower. This overall result shows that the coloured points are not necessary for tracking, as equivalent results can be obtained without using the colours. However, the colours can be used to improve the performance of the algorithm by ruling out false matches.

6.6.2 Gel phantom experiment

6.6.2.1 *Experiment*

A silicone gel cylinder was placed on the actuator plate of the hardware system described in detail in Chapter 8. Small coloured paper dots, were randomly applied to a patch on the cylinder, with an approximate density of 30 pts / cm². The gel cylinder was actuated at 100Hz, with an amplitude of 0.5mm.

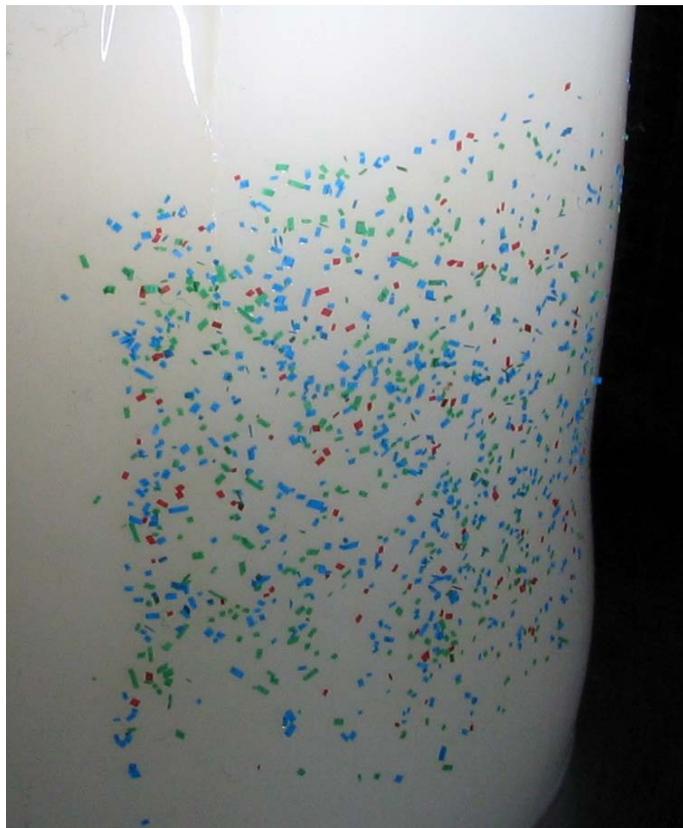
The coloured points were extracted by thresholding the separate colour channels of the colour images, as described in Chapter 9. One of the images is depicted in Fig. 6.8a. A number of the points were overlapping, due to the high density of the applied points, as can be seen in the close-up shown in Fig. 6.8b. No attempt was made to fix this problem, as it was desired to observe the robustness of the tracking procedure.

The 3-point Euclidean invariant signature tracking algorithm (Algorithm 6.2) was used to track the measured feature coordinates. Due to point overlaps, and the resulting uncertainty in centroid locations, a significant number of points were not successfully tracked. Of the approximately 700 points measured in the first frame (including spurious points), approximately 68% were successfully tracked through all 20 frames back to the first. The trajectories of the tracked points are shown in Fig. 6.9.

It is interesting to observe that despite the apparent haphazard placement of the points, as can be seen in Fig. 6.8b, the point trajectories appear fairly regular, especially in contrast to Fig. 6.6. The feature point locations are defined to be the centroid of the segmented regions, which acts as an averaging, or smoothing, operation that minimises the effect of errors in the segmentation process.



(a) Gel cylinder with coloured dots used for tracking experiment



(b) Close-up, where the irregularities and overlap in the dot placement can be seen

Figure 6.8: Gel cylinder used in tracking experiment

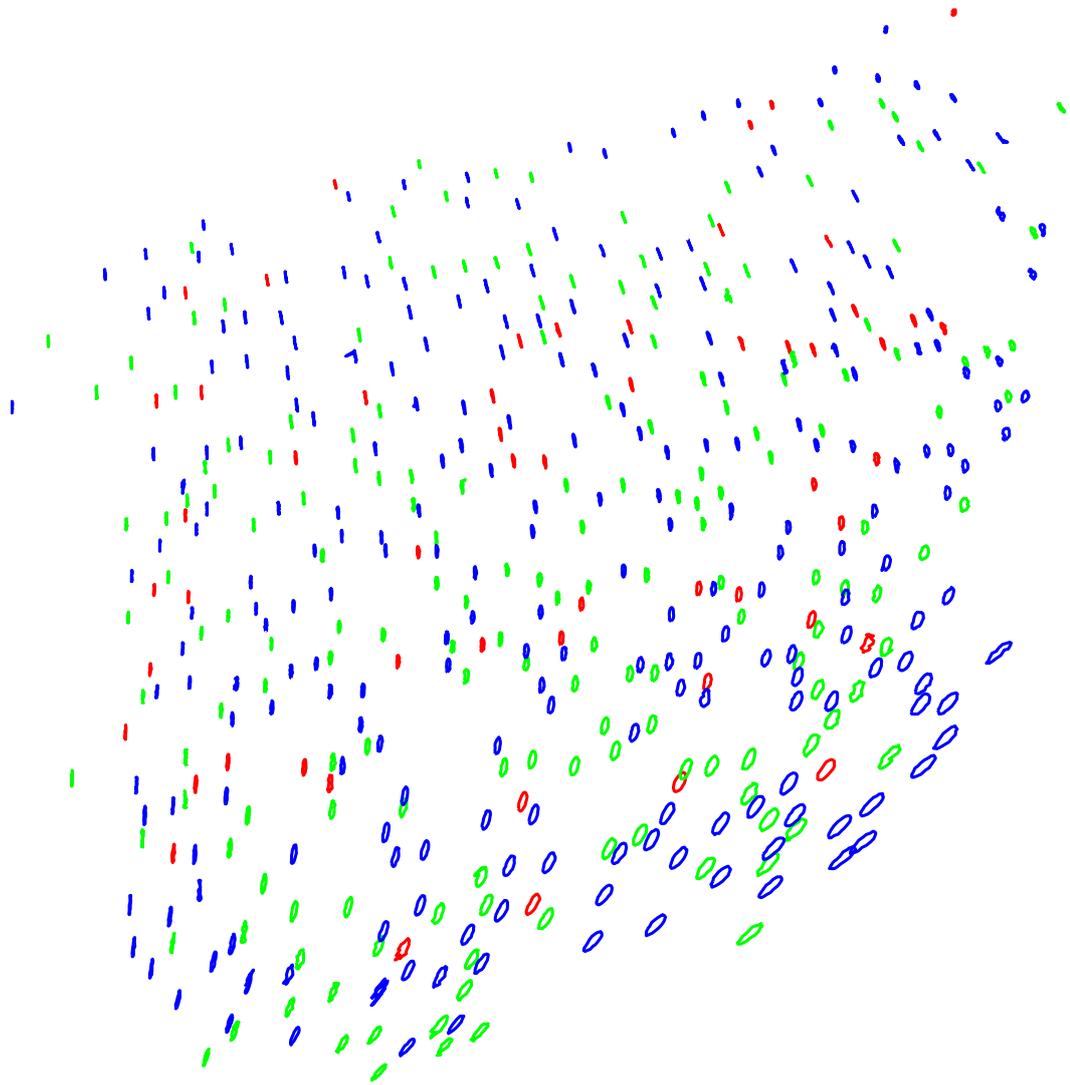


Figure 6.9: Trajectories of successfully tracked points in gel phantom experiment

6.6.2.2 *Summary*

The signature tracking algorithm was used to successfully track a high density of feature points on the actuated gel cylinder. Despite using irregularly shaped and overlapping points, approximately 68% of the points were successfully tracked through all 20 frames. It was also verified that the use of an Euclidean invariant signature was appropriate for the transformations induced in the images by the moving surface.

6.7 Summary

The problem of feature tracking in the DIET context was investigated in this chapter. Because of the potential high density of feature points on the breast surface, and the magnitude of the point motion compared with the point spacing, traditional tracking methods are inappropriate. Three tracking algorithms were presented using local Euclidean and similarity invariants to match key points between frames. These algorithms were used to successfully track features in experiments, and their relative performance was discussed.

The algorithms presented are examples of the more abstract invariant signature concept. The discrete signatures based on joint signatures could easily be extended to other transformation groups, such as the affine or projective group, however noise becomes a significant problem in these cases, due to the presence of skew and projective distortion.

The algorithms have been shown to be suitable for use in the DIET project, with different algorithms being suited to different conditions. If point spacing is large, nearest neighbour tracking, presented in Algorithm 6.1, should be used. If point spacing is small, and the motion relatively large, and coloured points are used, then the 3-point Euclidean invariant signature, presented in Algorithm 6.2 should be used, and if single-coloured points are used with small point spacing, the unordered similarity invariant signature should be used, presented in Algorithm 6.3.

Chapter 7

SEER: A 3D SURFACE RECONSTRUCTION ALGORITHM

7.1 Introduction and prior work

This chapter is concerned with the problem of 3D smooth surface reconstruction from image data. After the features have been tracked using one of the tracking algorithms described in Chapter 6, it is necessary to match the image point trajectories between cameras, so that the 3D position of each pair of corresponding points on the trajectories can be computed. The algorithm presented in this chapter takes as input point features, rather than point trajectories. The centroids of the trajectories are hence used as the input to the algorithm. In reconstructing the surface, corresponding points are automatically identified, and these correspondences can be used to reconstruct motion trajectories. This procedure is explained in more depth in Chapter 9.

Conventional 3D surface reconstruction methods fall loosely into two categories. There are volume-based methods, which use various image constraints to constrain the volume in space into which the imaged object falls, where the resulting volume is treated as the 3D representation of the object, e.g. [22, 23, 27]. These methods are fast and have many applications, particularly in computer graphics and augmented reality. A second 3D reconstruction method is dense stereo, e.g. [18, 33], where a dense set of correspondences are found by first identifying matching ‘key’ points, and then finding dense correspondences by searching along epipolar scanlines from each image point.

The interest points used in stereo algorithms are points in the image where the intensity information in a small neighbourhood exhibits some kind of irregular behaviour. For example, the Harris corner detector [15] finds image points which exhibit low self-similarity by correlating a small window around the point with shifted versions in each direction. Points for which all of these correlations are low are classed as interest points. Other common interest point detection algorithms include SUSAN [35] and SIFT [24]. Image point correspondences between the two images are typically found by identify-

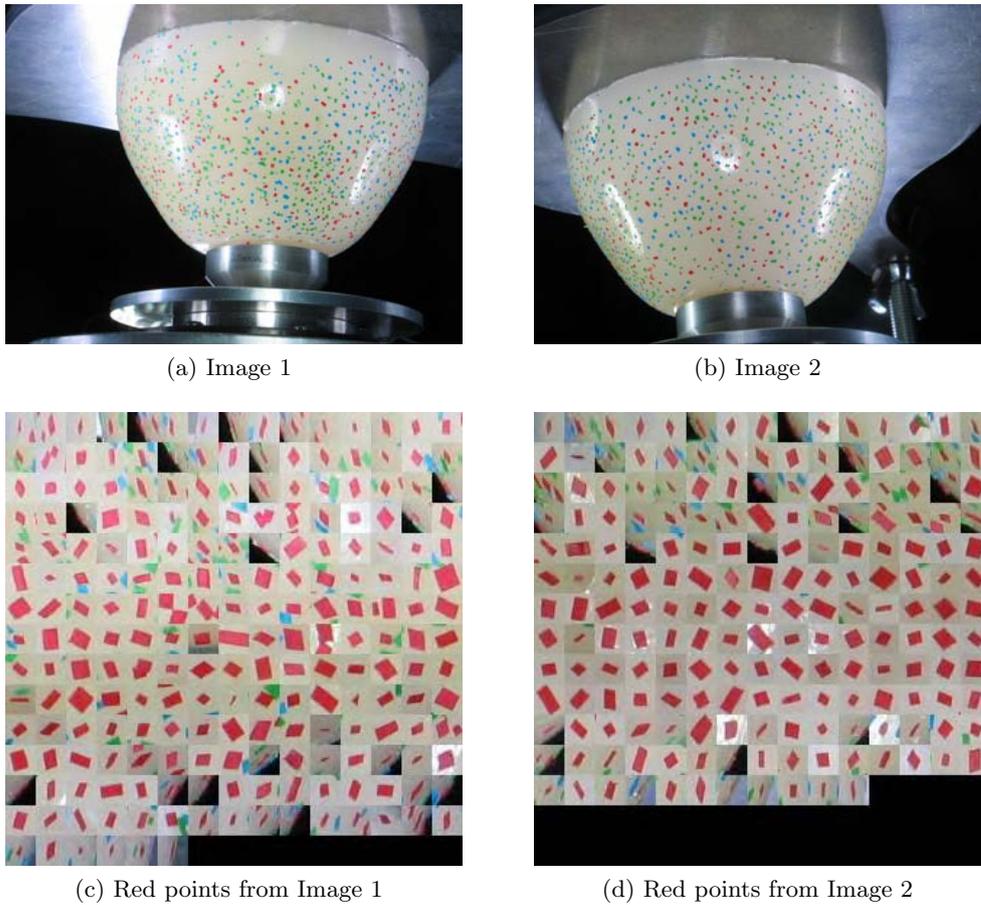


Figure 7.1: Views of a phantom from two different cameras, shown together with a small neighbourhood of each red point

ing feature points where the correlation, usually Normalised Cross-Correlation (NCC), between the neighbourhoods of each is high. In the case of SIFT, the detector computes a number of properties of the feature which should be approximately invariant to viewpoint changes, and these properties are used to match interest points.

The problem with using a feature point detector is that all of the features look more or less the same. Consider, for example, the images depicted in Fig. 7.1, from an experiment performed with a gel phantom, described later in Chapter 9. The red points have been extracted from each image and displayed underneath. The problem that the interest point detector / NCC approach is essentially trying to answer is ‘*Which of the red points from panel (c) matches which of the red points from panel (d)?*’ This is a very difficult question to answer, because, apart from random irregularities in shape, they all look essentially the same. If, in Fig. 7.1, colours weren’t used, the problem would be significantly worse.

If the cameras are calibrated, the correspondence problem is made appreciably easier, as the matches can be constrained to lie on corresponding image epipolar lines,

as will be discussed later. This reduces the number of candidate matches for each point to a more manageable number, in the case of the images in Fig. 7.1, each point has approximately 2-5 candidate matches to consider. It is still difficult, however, to establish which of the candidates is the true match.

Finally, in addition to stereo vision techniques, there are some methods which attempt to match point sets undergoing non-rigid deformations, such as TPS-RPM [6] and a neighbourhood-preserving relaxation approach [43]. These methods, however, don't work well with an appreciable amount of projective distortion, and also require a reasonably good initial transformation estimate to seed the nonlinear optimisation procedures used. They are also very computationally intensive. It is clear that without some more information about the problem, obtaining the reconstruction is very difficult.

The algorithm presented in this chapter solves a more specialised problem, that of reconstructing a smooth surface that has a dense set of self-similar features, i.e. the breast surface with coloured fiducials added. In other words, the problem considered is reconstructing a surface for which it is possible to extract the feature locations accurately, but difficult to solve the correspondence problem of identifying matching features between the two cameras as the features are very similar in appearance. For this problem, rather than using properties of the image intensity data to identify points, a geometrical approach is used.

7.1.1 Development Approach

The approach taken in this research is essentially geometric. Instead of trying to find unique image point correspondences, the candidate correspondences found by enforcing the epipolar constraint are used to construct a large cloud of points in 3D. The points reconstructed from correct correspondences will be the surface points, and the others will be essentially randomly distributed in space. The surface can be found by making use of extra information about the geometry of the surface. The surface is smooth, and the point features are relatively dense on the surface. Therefore, the surface points in space will fall on a smooth surface which looks locally like a plane, whereas the spatial configuration of the remaining points will be essentially random. The surface points can therefore be found by finding regions where a large number of points fall on a plane.

The overall algorithm is denoted "SEER", standing for Surface Extraction from Epipolar-constrained Reconstruction. The algorithm, while designed for use in the DIET project, is suitable for 3D surface reconstruction of any relatively smooth surface. It only requires a dense set of feature points imaged from two (or more) calibrated cameras, together with approximate knowledge of the average point density on the surface.

7.2 3D Reconstruction Algorithm

7.2.1 Epipolar point cloud construction

Consider images $\mathcal{I}, \mathcal{I}'$ taken from cameras with projection matrices \mathbf{P}, \mathbf{P}' . The camera poses induce an epipolar geometry, where a potential match $\mathbf{x}' \in \mathcal{I}'$ for a point $\mathbf{x} \in \mathcal{I}$ is constrained to lie on an *epipolar line* in \mathcal{I}' . The epipolar line has homogeneous coordinates \mathbf{l}' , such that $\mathbf{l}'^\top \mathbf{u} = 0$ for all points $\mathbf{u} \in \mathcal{I}'$ on the line. The epipolar line \mathbf{l}' corresponding to a given point \mathbf{x} is given by $\mathbf{l}' = \mathbf{F}\mathbf{x}$ where $\mathbf{F} \in \mathbb{R}^{3 \times 3}$ is a rank-2 matrix known as the fundamental matrix which can be computed directly from the two projection matrices \mathbf{P}, \mathbf{P}' , as described in §3.6.2.

All epipolar lines meet at a single point, called an epipole, which is the image of the optical centre of the other camera. In the absence of noise, points $\mathbf{x} \in \mathcal{I}$ and $\mathbf{x}' \in \mathcal{I}'$ satisfy the epipolar constraint if

$$\mathbf{x}^\top \mathbf{F}^\top \mathbf{x}' = 0. \quad (7.1)$$

A geometric interpretation of the epipolar constraint is that two points satisfy the epipolar constraint if the rays back-projected from the camera centres through the points meet in space. In reality, image point locations and the fundamental matrix itself are subject to measurement noise and so (7.1) is in general not satisfied for corresponding points. For a noisy image point measurement $\hat{\mathbf{x}}$, the corresponding point will not necessarily lie on the line $\mathbf{F}\hat{\mathbf{x}}$, but will fall within the envelope of a small range of epipolar lines. If measurement errors are small and the epipole is located outside of the image plane, this range of lines will be approximately parallel, and the epipolar constraint in the presence of noise can be written

$$\mu \left| \mathbf{x}^\top \mathbf{F}^\top \mathbf{x}' \right| < \delta_{\text{ep}} \quad (7.2)$$

where $\mu > 0$ is a scale factor such that the quantity $\mu \left| \mathbf{x}^\top \mathbf{F}^\top \mathbf{x}' \right|$ represents the perpendicular distance between the line $\mathbf{F}\mathbf{x}$ and the point \mathbf{x}' . The threshold δ_{ep} can be determined heuristically. However, if a Gaussian noise model is assumed for image point measurements, it can be estimated analytically or by Monte Carlo simulation [16]. An example of a Monte Carlo simulation to estimate δ_{ep} is given in the case study in Chapter 9.

The first step of the reconstruction procedure is to find all pairs of points in the two images $\mathcal{I}, \mathcal{I}'$ that satisfy the epipolar constraint (7.2). An estimate of the 3D spatial point \mathbf{X} giving rise to the measurements \mathbf{x}, \mathbf{x}' can then be constructed by *triangulation*, finding the point in space closest, to the two rays back-projected from the optical centres of the two cameras through the image points. The Direct Linear Transformation (DLT) algorithm [16] is used to triangulate pairs of points.

All pairs of points satisfying (7.2) are triangulated in this fashion, yielding a point

cloud, comprising the imaged surface, and a large number of outlying points from the false correspondences. This process is illustrated in Fig. 7.2. Denote this point cloud by \mathcal{M} .

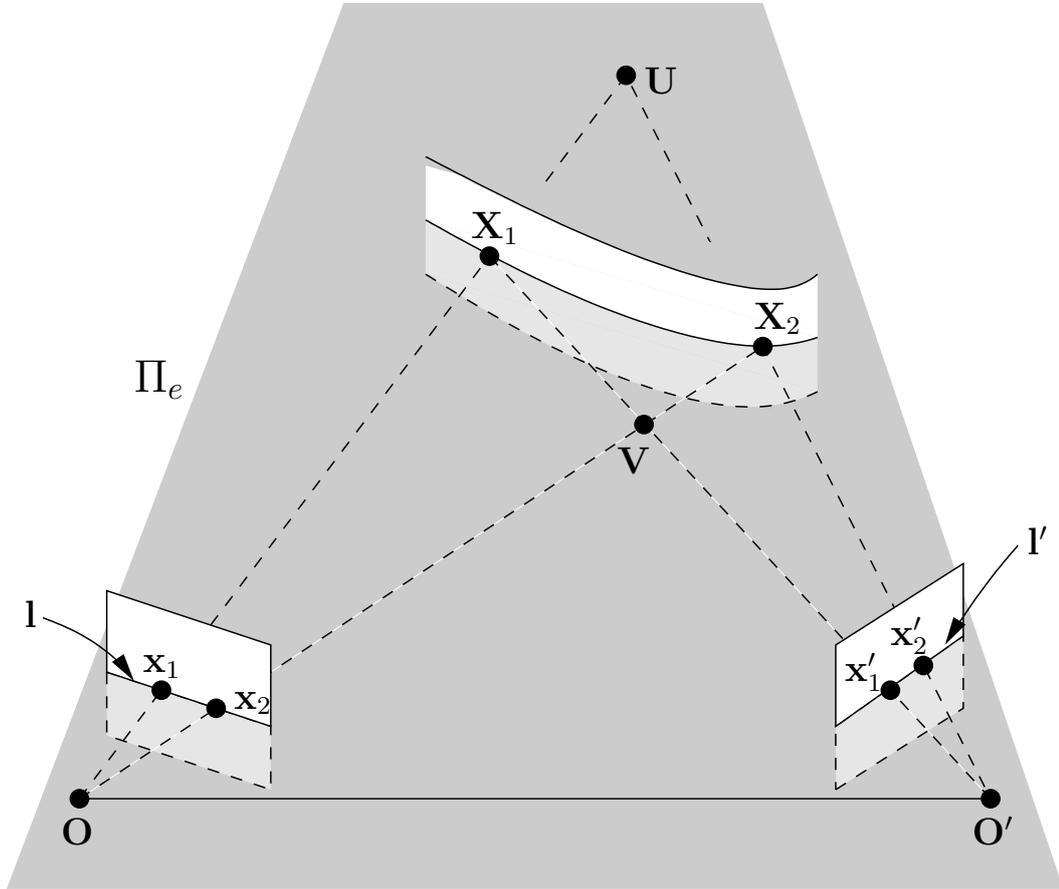


Figure 7.2: Epipolar constraint and point cloud formation. Any choice of two points on l and l' will satisfy the epipolar constraint (§3.6.1). In this image, the two world points $\mathbf{X}_1, \mathbf{X}_2$ generate two image points in each image. Each pair of points $(\mathbf{x}_1, \mathbf{x}'_1)$, $(\mathbf{x}_1, \mathbf{x}'_2)$, $(\mathbf{x}_2, \mathbf{x}'_1)$, $(\mathbf{x}_2, \mathbf{x}'_2)$ satisfies the epipolar constraint, and the resulting constructed points in space are the true surface points $\mathbf{X}_1, \mathbf{X}_2$, together with two additional points \mathbf{U}, \mathbf{V} resulting from incorrect correspondences.

7.2.2 Parameters

Denote the set of fiducial points on the surface \mathcal{S} by \mathcal{P} . Suppose that the mean density of points on the surface \mathcal{S} is ρ . If the points exhibit total randomness, the distribution can be modelled by a homogeneous Poisson process, and the number of points in a region of surface area A will be Poisson distributed with mean ρA . Define the three related quantities $B_r(\mathbf{X}), B_r(\mathbf{X}, \mathcal{S}), B_r(\mathbf{X}, \mathcal{P})$ as follows. Let

$$B_r(\mathbf{X}) = \{\mathbf{Y} \in \mathbb{R}^3 \mid \|\mathbf{X} - \mathbf{Y}\|_2 < r\}$$

be the open radius r ball in \mathbb{R}^3 centred at \mathbf{X} . Let

$$B_r(\mathbf{X}, \mathcal{S}) = \{\mathbf{Y} \in \mathcal{S} \mid \|\mathbf{X} - \mathbf{Y}\|_2 < r\}$$

be the region of \mathcal{S} within distance r of \mathbf{X} , which will approximately resemble a planar region with a circular boundary, and let

$$B_r(\mathbf{X}, \mathcal{P}) = \{\mathbf{P} \in \mathcal{P} \mid \mathbf{P} \in B_r(\mathbf{X}, \mathcal{S})\}$$

be the set of points in \mathcal{P} contained in $B_r(\mathbf{X}, \mathcal{S})$. Note that the distance is not the geodesic distance on the manifold \mathcal{S} , but the Euclidean distance in the underlying space \mathbb{R}^3 . If r is sufficiently small that $B_r(\mathbf{X}, \mathcal{S})$ is approximately planar, then the mean number of points in $B_r(\mathbf{X}, \mathcal{P})$ is approximately $n_B = \rho\pi r^2$. Conversely, for a specified n_B , choosing a radius r defined:

$$r = \sqrt{\frac{n_B}{\rho\pi}} \quad (7.3)$$

will yield regions $B_r(\mathbf{X}, \mathcal{P})$ that contain a mean number of points n_B , so long as the surface is approximately planar at this radius.

To quantify the extent to which the regions $B_r(\mathbf{X}, \mathcal{S})$ are planar, let d_c be the maximum perpendicular deviation in $B_r(\mathbf{X}, \mathcal{S})$ from the tangent plane $T_{\mathcal{S}}(\mathbf{X})$ to \mathcal{S} at \mathbf{X} . This is more conveniently represented by the dimensionless quantity

$$\kappa = d_c/r = \sin \theta \quad (7.4)$$

where θ is the angle between the tangent plane $T_{\mathcal{S}}(\mathbf{X})$ and a line drawn between \mathbf{X} and a point with perpendicular distance d_c from $T_{\mathcal{S}}(\mathbf{X})$ on the boundary of $B_r(\mathbf{X}, \mathcal{S})$. This κ is used, rather than a more conventional measure of curvature, such as the maximum principal curvature, because it is easily conceptualised. In a region of radius r , κ is simply the maximum permitted deviation from the tangent plane, divided by the r . These entities are more easily visualised in a picture, as shown in Fig. 7.3a. Referring to Fig. 7.3a, the minimum radius of curvature r_c is given by

$$r_c = \frac{r}{2 \sin \theta} = \frac{r}{2\kappa} \quad (7.5)$$

7.2.3 RANSAC tangent plane estimation

Given a point $\mathbf{X} \in \mathcal{S}$, The RANSAC algorithm [12], can be used to construct an estimate $\hat{T}_{\mathcal{S}}(\mathbf{X})$ of the tangent plane to \mathcal{S} at \mathbf{X} . Let \mathcal{B} be the set of points neighbouring \mathbf{X}

$$\mathcal{B} = \{\mathbf{Y} \in \mathcal{M} \mid \mathbf{Y} \in B_r(\mathbf{X}), \mathbf{Y} \neq \mathbf{X}\}.$$

$\hat{T}_{\mathcal{S}}(\mathbf{X})$ is computed by fitting a plane robustly to \mathcal{B} , rejecting outlying points that are not on the surface as follows. Two points are randomly sampled from \mathcal{B} and a plane constructed through these two points and \mathbf{X} . Allowing for the curvature of the surface as described in §7.2.2, and a noise threshold ϵ , the region depicted in Fig. 7.3b about this estimated plane $\hat{T}_{\mathcal{S}}(\mathbf{X})$ is searched for inlying points. This process is repeated until the probability of having sampled two inliers from \mathcal{B} is greater than $p = 0.99$. The number of iterations required [16] to achieve this goal is defined:

$$N_{\text{iterations}} = \left\lceil \frac{\log(1-p)}{\log(1-w^2)} \right\rceil \quad (7.6)$$

where w is the proportion of inliers in \mathcal{B} , which is not required to be known in advance; w can be estimated at each iteration by taking the largest number of inliers found and dividing by the number of elements in \mathcal{B} , and $N_{\text{iterations}}$ updated by (7.6). The largest set of inliers resulting from this process is finally used to compute the estimate the plane $\hat{T}_{\mathcal{S}}(\mathbf{X})$ by least squares, minimising the sum of the squares of perpendicular deviations from the plane (not an algebraic error). The homogeneous coordinates of the set of inlying points are normalised to be of the form $\mathbf{X}_i = [X_i, Y_i, Z_i, 1]^T$, $i = 1, \dots, n$ and then stacked into the $n \times 4$ matrix $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n]^T$, and homogeneous coordinates for $\hat{T}_{\mathcal{S}}(\mathbf{X})$ are computed by solving the constrained linear least squares problem

$$\arg \min_{\pi = [\pi_1, \dots, \pi_4]^T} \|\mathbf{X}\pi\|^2 \quad \text{subject to} \quad \pi_1^2 + \pi_2^2 + \pi_3^2 = 1$$

which can be solved by standard SVD based methods [14].

7.2.4 Reconstruction algorithm

For each point $\mathbf{X} \in \mathcal{M}$, the tangent plane described in §7.2.3 is fitted, with the exception of points whose neighbourhood does not contain a sufficient number of points. Note that the tangent plane fitting algorithm will still work if the point \mathbf{X} is not on the surface \mathcal{S} , however the resulting fitted plane will be essentially random. An (undirected) adjacency graph \mathcal{G}_A with the points of \mathcal{M} as nodes is constructed, whereby points \mathbf{X} and \mathbf{Y} are considered to be connected if \mathbf{X} is an inlier to the tangent plane fit of \mathbf{Y} or vice versa.

The next step is to prune the graph \mathcal{G}_A so that points are only considered adjacent if their normals are suitably close. First, all edges involving nodes for which it was not possible to fit a tangent plane are discarded, as there is no estimate of their normal. Taking into account the allowable curvature of the surface as described in §7.2.2, for two points \mathbf{X} and \mathbf{Y} , separated by distance s the maximum allowable angular deviation

α between their normals is given by

$$\cos \alpha = \frac{2r_c^2 - s^2}{2r_c^2} \quad (7.7)$$

as depicted in Fig. 7.3c. Assuming unit normals, the angle β between the two normals is $\cos \beta = |\mathbf{n}_\mathbf{X} \cdot \mathbf{n}_\mathbf{Y}|$. Therefore, the new condition for two points \mathbf{X} and \mathbf{Y} to be adjacent is defined:

$$|\mathbf{n}_\mathbf{X} \cdot \mathbf{n}_\mathbf{Y}| > \frac{2r_c^2 - \|\mathbf{X} - \mathbf{Y}\|^2}{2r_c^2}, \quad (7.8)$$

and the graph \mathcal{G}_A is pruned accordingly, with edges not satisfying (7.8) being removed from the graph.

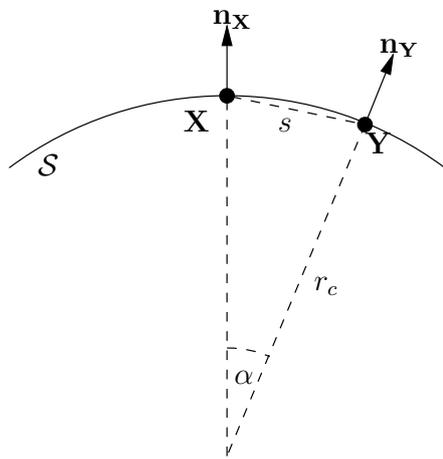
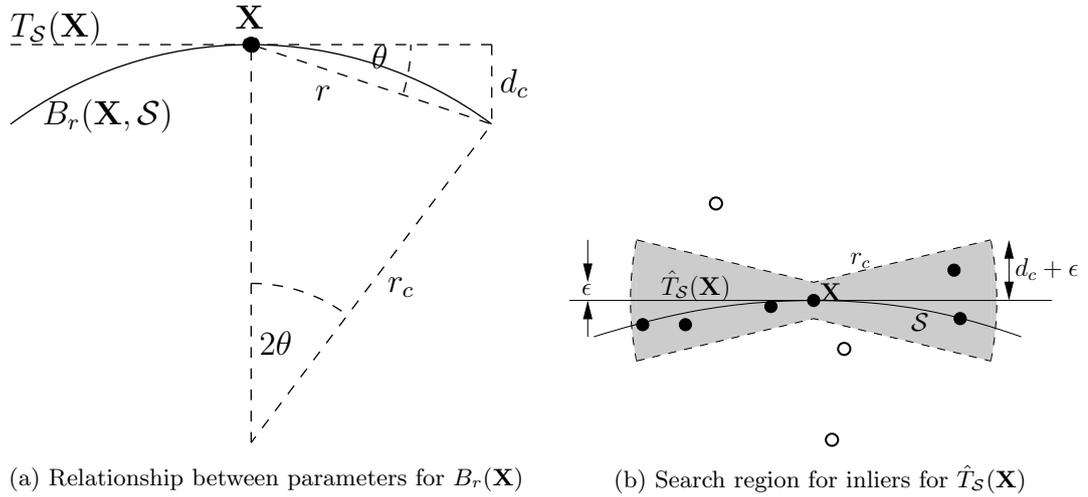
Next, a connected component analysis is performed on \mathcal{G}_A by depth-first search, and the reconstructed surface is defined to be the largest connected component of \mathcal{G}_A . The final step is to remove outliers. Any outliers lie close to the surface, so this is done by fitting a plane using RANSAC to the neighbourhood $B_r(\mathbf{X})$ of each point on the reconstructed surface, and marking the point as an outlier if it is not an inlier to the fitted plane. Note that, in contrast to the procedure in §7.2.3, the plane is not constrained to pass through \mathbf{X} .

7.2.5 Implementation Notes

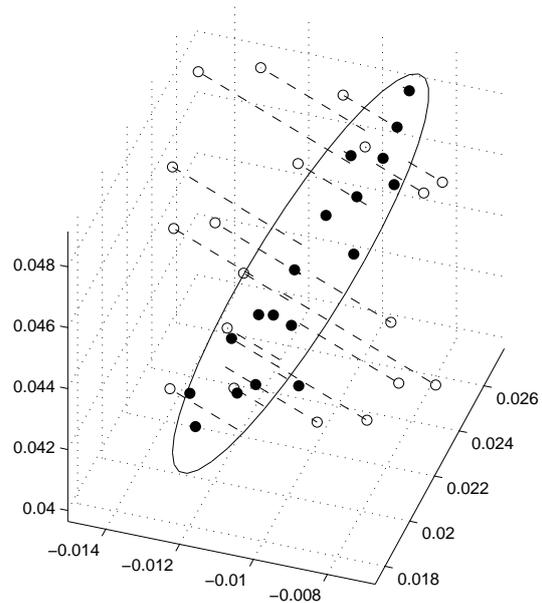
There are a number of parameters referred to in the previous sections. The parameters n_B and κ are tunable. From experiments, reconstruction appears to work well for $n_B \approx 15$ and $\kappa \approx 0.1$. With these numbers in mind, if the application allows the feature points to be applied to the surface, then they should be applied so that a region with about n_B points looks, at least by eye, to be approximately planar.

An estimate of ρ is required. Accuracy is not critical as an estimate of the nature of “about 10 points per cm^2 ” is sufficient. The reconstruction noise threshold ϵ will depend on the cameras and configuration used. If results are unsatisfactory, the parameters n_B and κ can be tuned accordingly. The effect of the tuning parameters n_B, κ , the range of densities ρ that are suitable, and the required accuracy of the estimate ρ_{est} will be investigated further in §7.3.1.

To prevent the point cloud from being too dense near the surface, it is important that the epipolar matching threshold δ_{ep} is as low as possible, to minimise the number of correspondences satisfying the epipolar constraint, and hence minimise the total number of points in the cloud. Another means of reducing the density of the point cloud, and also the reconstruction speed is to use, for example, points of different colours. If three colours were used, and points only considered to satisfy the epipolar constraint if their colour was the same, then the number of potential correspondences will be approximately one third of the number if colour were not taken into account.



(c) Maximum deviation of α allowed between normals of points \mathbf{X} , \mathbf{Y} , distance s apart



(d) A typical search region - the detected inliers are shown as filled circles, and outliers as open circles. Note that the inliers fall approximately on a plane, while the outliers are essentially randomly distributed

Figure 7.3: Constraints for the plane fitting procedure

Point clouds are shown from a laboratory example where one and three colours are used in Fig. 7.4. The difference in cloud density can clearly be seen, while the density at the surface remains approximately the same.

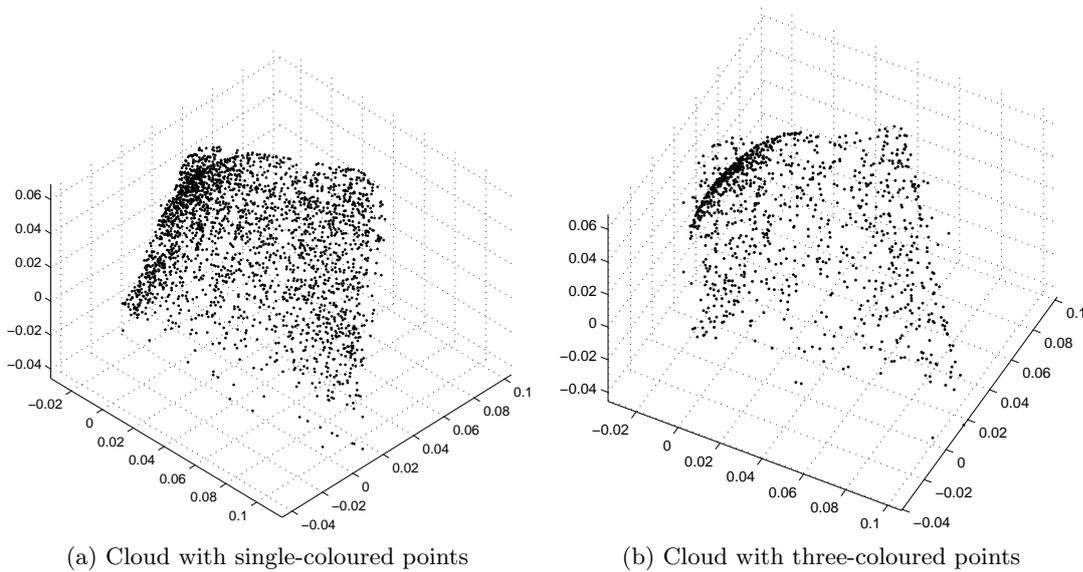


Figure 7.4: Epipolar point clouds where one or three colours are used

7.3 Case study

7.3.1 Computer Simulation

The test case for the algorithm was a hemisphere with randomly distributed points generated by a homogeneous Poisson process on the surface. The hemisphere had radius 5cm, and synthetic images were taken from five cameras evenly spaced around the hemisphere. The cameras were evenly positioned on a circle of radius 20cm, elevated 20cm above the base of the hemisphere, pointing at the centre of the base of the hemisphere. Five cameras were used as this was the minimum required to have each region of the hemisphere viewed by two overlapping cameras. The camera matrices were defined with parameters such that the image of the hemisphere approximately filled a 1600×1200 pixel region. The surface was reconstructed by performing the reconstruction pairwise for pairs of adjacent cameras, and then combining the resulting surfaces for the final result. Experiments were performed with varying densities of points on the surface, using either one or three colours of points, and with different levels of measurement noise. The effect of the tuning parameters n_B, κ are examined within this context.

Different stages of the reconstruction process are shown in Fig. 7.5. For the illustrated simulation, the following parameters were used, $\rho = 10$ pts/cm², $\kappa = 0.1$,

$n_B = 15$, $\epsilon = 0.5$ mm, and $\delta_{ep} = 1$ pixel. Colour information was not used, and Gaussian noise of standard deviation $\sigma = 0.5$ pixels in each coordinate was added to the image coordinates. Fig. 7.5, (a) shows surface points as seen by one camera, (b) shows the point cloud resulting from epipolar correspondences between two cameras, (c) shows the extracted surface from the cloud in (b), and (d) shows the reconstruction from the five pairs of cameras combined.

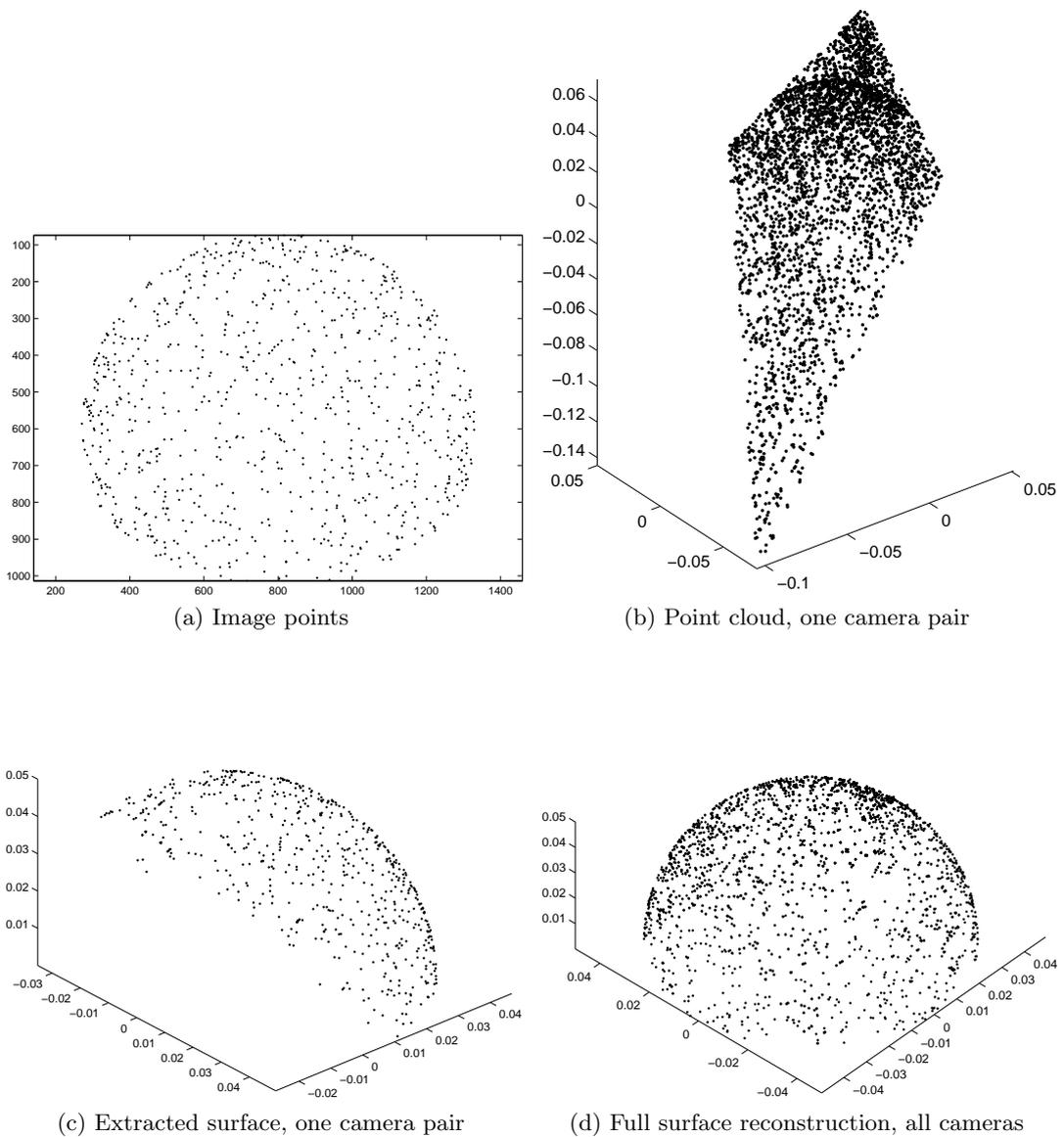


Figure 7.5: Surface reconstruction process for a 5cm radius hemisphere with points randomly distributed with a mean density of $\rho = 10$ pts/cm²

ρ (pts/cm ²)	n_{colours}	r (mm)	r_c (cm)	p_{correct}	p_{surf}
2	1	15	7.7	98	20
5	1	9.8	4.9	96	89
10	1	6.9	3.5	94	87
15	1	5.6	2.8	90	87
20	1	4.9	2.4	88	86
25	1	4.4	2.2	89	83
2	3	15	7.7	98	42
5	3	9.8	4.9	99	87
10	3	6.9	3.5	97	88
15	3	5.6	2.8	96	89
20	3	4.9	2.4	96	89
25	3	4.4	2.2	96	88
2	1	15	3.9	94	87
2	3	15	3.9	97	86

Table 7.1: Surface reconstruction for varying point density ρ . The last two rows have $\kappa = 0.2$, the rest have $\kappa = 0.1$

7.3.1.1 Performance under varying point density

An experiment was performed varying the density of points on the surface between $\rho = 5.0$ and $\rho = 25.0$ pts/cm² while holding the other parameters constant at typical¹ values: $\kappa = 0.1$, $n_B = 15$, $\epsilon = 0.5$ mm, and $\delta_{\text{ep}} = 2$ pixels. First, points of one, and then points of three colours were used. Gaussian noise of standard deviation $\sigma = 0.5$ pixels in each coordinate was added to image coordinates. The estimated density ρ_{est} was set to be the same as the actual density used. For each case, the percentage of the true surface points reconstructed p_{surf} , and the percentage of correctly constructed surface points inliers p_{correct} , were computed. The results are shown in Table 7.1.

There are a few interesting features in Table 7.1. First, p_{surf} is never above 90%. The points on the surface that don't get reconstructed are mostly near the boundary of the surface and no attempt has been made in this paper to compensate for these edge effects. As the density of the points increases, the minimum allowable radius of curvature r_c decreases, see (7.5). For the lowest density considered, $\rho = 2$ pts/cm², r_c is too large, greater than the radius of the hemisphere, and hence less than half the hemisphere was reconstructed, both in the one-colour and three-colour cases. This situation was compensated for by increasing κ to 0.2, as shown in the last two rows of the table.

At the other end of the scale, at $\rho = 25$ pts/cm², r_c is relatively small at 2.2 cm; the algorithm is allowing for the presence of more curvature than is present², however performance is still satisfactory. Although there are between 2 and 12% outliers in each

¹Typical values for the DIET project

²Note that the true radius of curvature is a constant 5 cm throughout the hemisphere

case, these outlying points were classified as inliers in the plane-fitting step because they are indistinguishable from true surface points under the noise and curvature conditions. These outliers arise from points being very close together; using three colours to reduce the number of correspondences significantly increased the proportion of inliers. Other ways of accomplishing this task are improving the calibration and camera system so that δ_{ep} can be decreased, or introducing regularity into the point distribution to minimise the number of closely adjacent neighbours.

These experiments show that the algorithm works well at a reasonably large range of point densities, as low as 2 pts/cm² on a 5 cm radius hemisphere.

7.3.1.2 *Effect of parameters n_B, κ, ρ_{est}*

It is difficult to determine the effect of the different tuning parameters in isolation. To analyse their affect on the reconstruction procedure, experiments were performed under typical conditions, and each parameter varied individually to determine its effect. The results are shown in Table 7.2. The density was held constant at $\rho = 10$ pts/cm², one colour was used, and noise was added, as in the previous section.

Table 7.2 shows that the reconstruction is not particularly sensitive to the choices of the parameters, except perhaps for κ . The value of n_B needs to be chosen sufficiently high that there are almost always enough points in a given neighbourhood to construct the tangent plane. From the table, it seems that any value over 10 seems adequate. It does need to be sufficiently low that the point neighbourhoods containing n_B points are still reasonably planar.

The algorithm is most sensitive to the choice of κ , as this determines the amount of curvature allowed. As κ gets higher the chances of erroneously fitting the wrong tangent plane increase, and more outliers get marked as inliers. The choice of κ is somewhat determined by the surface being imaged. The denser the points, the more planar the surface, and the lower κ can be set.

Errors in ρ_{est} also seem to make little difference in performance, the results are fairly consistent until it reaches 20 pts/cm², double the actual density. Therefore, a very rough estimate of ρ is all that is required.

7.4 Discussion and Conclusion

The approach taken, finding inliers by fitting planes to approximately planar regions is not the only one available. Other options could be and were considered. For instance, rather than fitting a plane, fitting a more general surface in the inliern search procedure was considered. Possible surfaces, in order of number of parameters, include spheres, ellipsoids, or a general quadric surface. There are a number of potential disadvantages of proceeding in this manner. Fitting second order surfaces is slower, especially if

n_B	κ	ρ_{est} (pts/cm ²)	p_{correct}	p_{surf}
5	0.1	10	95	23
10	.	.	94	84
15	.	.	92	87
20	.	.	90	88
25	.	.	93	89
15	0.05	10	89	08
.	0.10	.	93	88
.	0.15	.	85	88
.	0.20	.	82	88
.	0.25	.	75	88
.	0.50	.	53	88
15	0.1	5	88	88
.	.	7.5	93	85
.	.	10	92	86
.	.	12.5	94	87
.	.	15	95	83
.	.	20	95	63

Table 7.2: Results for computer simulation varying $n_B, \kappa, \rho_{\text{est}}$

nonlinear least squares is used. The least squares procedure is likely to be badly conditioned, because neighbouring points will be nearly coplanar. It is also difficult to constrain the least squares procedure to produce a surface of the right kind, (e.g. a sphere or ellipsoid), and to also have low curvature.

When using the algorithm, there are a number of factors influencing performance that can be traded off. A more accurate surface can be fitted by increasing the density of the points on the surface. However, this approach results in more epipolar correspondences, and hence, a larger point cloud and longer processing time to extract the surface. The number of correspondences can be reduced by either increasing the accuracy of the calibration system to reduce the epipolar threshold δ_{ep} , which is not always possible, or alternative means such as introducing a number of colours to the points.

In §7.3.1 the effect of the different tuning parameters was investigated, and the viability of the algorithm under a variety of different conditions. Importantly, it was demonstrated that values for the tuning parameters are not critical; successful reconstructions were performed for a wide range of parameter values. This result is important because it illustrates the robustness of the approach for this problem. Thus, in application, the tuning parameters can be set at typical values, ($n_B = 15$ and $\kappa = 0.1$), leaving fewer parameters ($\rho_{\text{est}}, \epsilon, \delta_{\text{ep}}$) to set for each reconstruction.

SEER has been used to successfully reconstruct surfaces both in computer simulation and in laboratory experiments. The algorithm was designed specifically for the DIET breast cancer screening project, however it shows promise for other applications requiring point-based surface reconstruction.

Chapter 8

EXPERIMENTAL DIET SYSTEM

8.1 Introduction

The two main tasks required to be performed by the experimental system are actuation and image capture. The actuator is required to mechanically actuate (vibrate) the breast sinusoidally at a specified amplitude and frequency. The image capture component is required to be capable of capturing still images of the moving breast at arbitrary phase differences from the actuator motion. Because of the frequency, and hence speed, of actuation, very high frame rates are required if the motion is to be captured continuously by, for instance, a high speed video camera. For example, if actuation is at 100Hz, then to capture 20 frames over one cycle would require a frame rate of 2000 frames per second. Standard consumer-grade movie cameras capture at a frame rate of approximately 25 frames per second. Cameras capable of achieving the required frame rate are very expensive, and have comparatively low resolution. Another inherent difficulty in using an array of high speed cameras is synchronising the frame capture of the different cameras.

Because the breast is actuated sinusoidally, the steady state motion of the surface is periodic at the actuation frequency. Therefore, the image capture can be solved by using a strobe light running at the same frequency as the actuator to obtain static images of the breast. With the strobe running, the exposure time of the camera can be as long as necessary. For the system described in this chapter, the cameras had exposure times of one second, yet the images were still crisp.

This chapter describes the experimental system used for image capture, which, with only small modifications is also suitable for clinical experiments. The system is a modular design, allowing for different components to be upgraded or replaced if required.

8.2 System components

8.2.1 Overview

The major components of the system are shown in Fig. 8.1. The system is broken down into the following distinct components:

dSpace Real time control of the actuator, and generation of the strobe signal is performed by a system running on the dSpace real-time hardware platform

Control Desk Control Desk software provides an interface to real-time adjustable dSpace settings, such as actuator frequency and amplitude

Camera server The camera server is a piece of software which communicates with dSpace via Control Desk and controls the cameras. The server coordinates dSpace settings changes, such as changes in strobe light phase delay, with image capture

Control Desk GUI The dSpace settings, such as strobe phase delay, operation frequency, and amplitude can also be controlled manually via a graphical interface developed in Control Desk. The main purpose of the GUI is to provide immediate visual feedback of the system performance and state

Camera client Job requests, which consist of parameters such as number of frames, frequency, and amplitude are sent to the camera server over a network connection by the camera client software. The camera client software is the means of user communication with the camera server

Actuator The actuator is a voice-coil actuator driven by a signal from dSpace, amplified by a power amplifier

Cameras The cameras are consumer-level Canon Powershot G5 cameras

Strobe The strobe lights are custom-built LED ring flashes retrofitted to each camera controlled by a simple TTL signal from dSpace

Each of these system components is described in more detail in the following sections.

8.2.2 Hardware platform

The hardware platform used for the control of the actuator and strobe is dSpace. dSpace was used as it is convenient for development, models can be built in Simulink and compiled directly to run on the dSpace module. The sample rate used was 10kSamples/second, giving a sample time of $100\mu\text{s}$, which is more than fast enough for the tasks required.

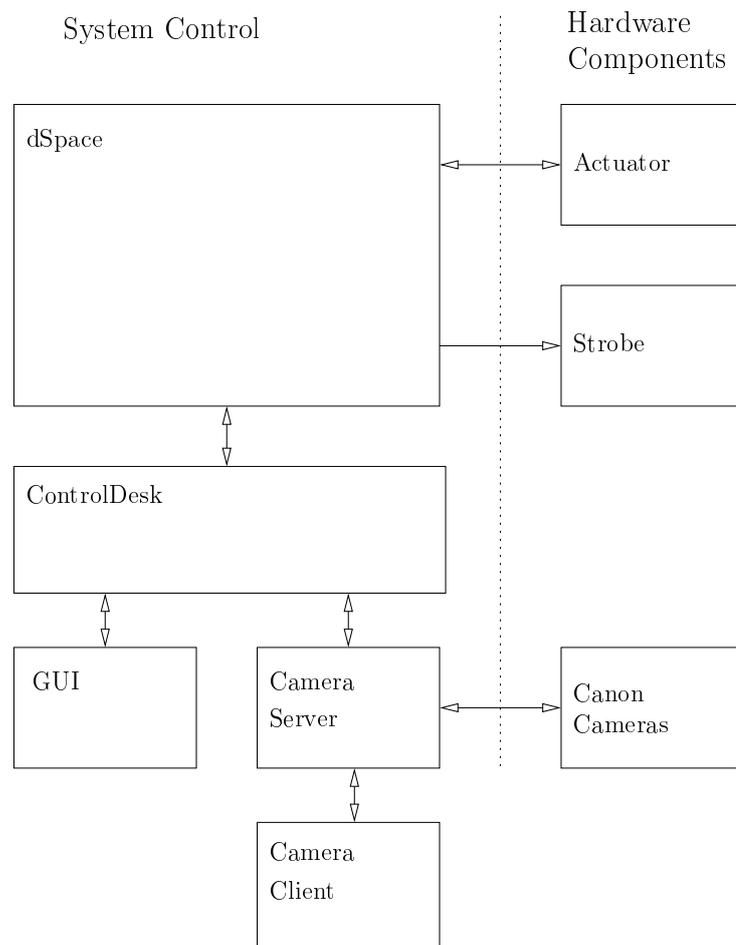


Figure 8.1: System diagram of the experimental setup showing the major components

8.2.3 Actuator

The actuator is a voice-coil type actuator, which was previously custom-built for the DIET project as a final year mechanical engineering project. The actuator is driven by a sinusoidal input, amplified by a 160W power amplifier. Vertical position feedback of the actuator plate is provided by an LVDT (linear variable differential transformer). The actuator, shown actuating a gel phantom, and four of the cameras, is shown in Fig. 8.2

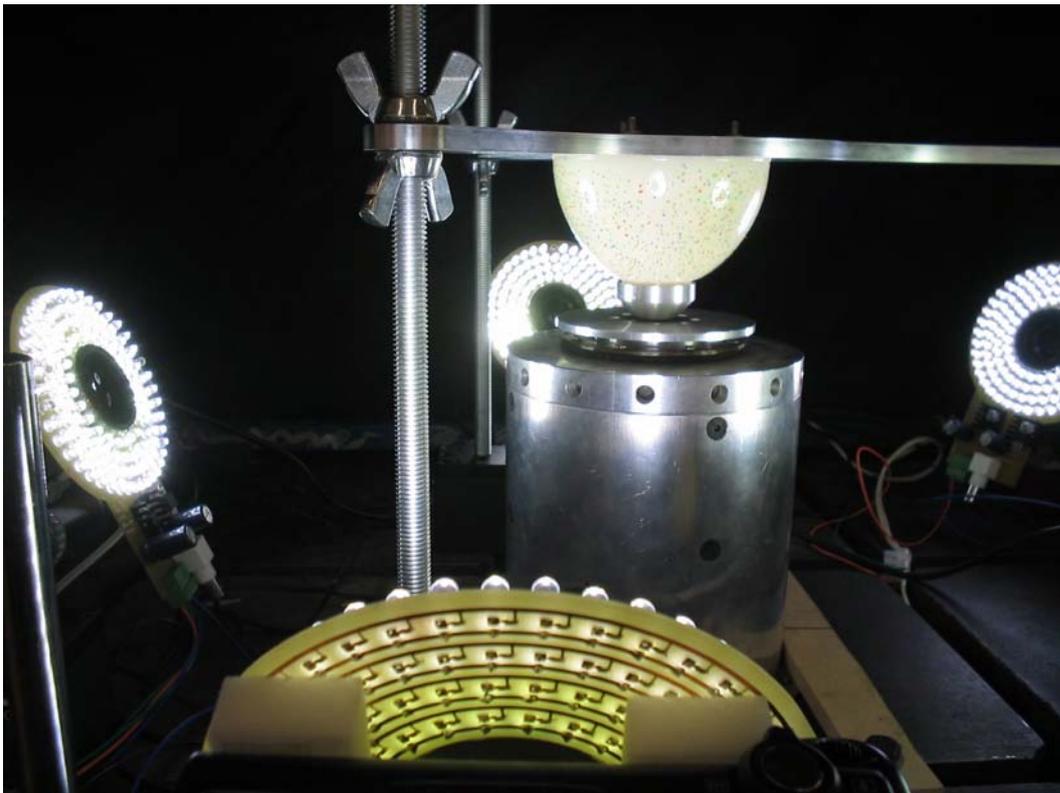


Figure 8.2: Actuator with gel phantom. Four of the five cameras, mounted with LED ring flashes can also be seen

8.2.4 Cameras

The cameras used are standard consumer Canon Powershot G5 digital cameras. For the majority of the experiments, the cameras were operated at two megapixels (1600×1200 pixels), rather than the full five megapixels the cameras can provide. The lower resolution was used because the accuracy obtained at the lower resolution is sufficient for DIET, and the increase in computation required for image processing the larger images was not deemed worthwhile. The reason these cameras are used, rather than entry level digital cameras, is the quality of the lens. There is very little radial distortion present in the images, and hence the cameras can be accurately calibrated using a linear

camera model (§5). Canon cameras were used because of the openly available software development kit (SDK), which enabled the development of the camera server software. Communication with the cameras is via a USB1.0 (Universal Serial Bus) connection.

8.2.5 Strobe: LED ring flashes

Four of the cameras can be seen in Fig. 8.2, each fitted with a custom-built array of high-brightness white LEDs (Light Emitting Diodes). The flashes are all connected to a single control signal which simply turns the LEDs on when high, and off when low. To use the LEDs as a strobe, a stream of narrow pulses, synchronised with the LVDT signal is used as the input. A pulse with a 2% duty cycle was used as the strobe signal to provide uniform lighting levels across different frequencies of operation.

8.3 Control software

8.3.1 dSpace controller

The two main tasks performed by the dSpace system are providing the actuator input signal and generating the synchronised strobe signal. The actuator signal is a constant-frequency sinusoid, with PID amplitude control via feedback from the LVDT position sensor. The PID controller is discussed in §8.3.1.1.

The strobe signal for the ring flashes is synchronised to the incoming LVDT signal, and delayed by a variable phase delay. Generation of the strobe signal is discussed in §8.3.1.2. The dSpace system is configured to allow real-time adjustment of certain parameters. The actuation frequency and amplitude, and the phase delay can all be adjusted during operation of the system.

There are two interfaces which allow adjustment of the experimental parameters. The first is a GUI, implemented in ControlDesk, as shown in Fig. 8.3, which allows visual feedback of system behaviour, as well as runtime modification of the parameters. The GUI controls are simple. The plot window shows the LVDT signal and the strobe pulse signal. There are two important panels of controls, the Actuator Control panel and the LED Lighting Control panel, as can be seen in the bottom half of Fig.8.3. The actuator controls allow the frequency and amplitude to be set, as well as showing the measured amplitude (in red). There is also a checkbox allowing the actuator to be temporarily switched off. The LED lighting control allows the phase delay relative to the LVDT signal, measured as the fraction of a full cycle, to be specified, as well as the duty cycle of the pulse. The duty cycle is the fraction of a period that the LED is on for.

The second interface is via a python server running in Control Desk, which is connected to by a socket connection over TCP/IP, allowing parameters to be changed

programmatically by a client anywhere on the network. Changes in parameter values sent over the socket connection are immediately reflected in the GUI.

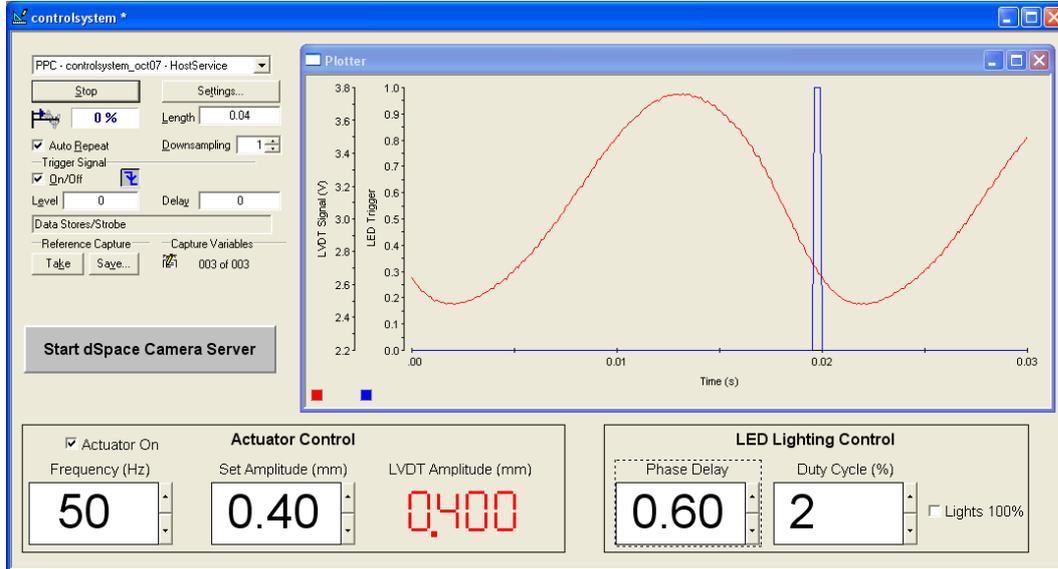


Figure 8.3: Control Desk GUI used for observation and parameter adjustment. The plot shows the LVDT measurement and the strobe pulse

8.3.1.1 Actuator amplitude control

The amplitude of the actuator motion is controlled by a simple PID controller, depicted in Fig. 8.4. The LVDT provides an analog measurement of the vertical position of the actuator plate, which is converted to a digital signal by one of the dSpace ADCs. The block H_{amp} converts this signal into an amplitude in mm. This is done by continuously buffering 50ms (500 samples) of measurements in a shift register and taking half the difference of the maximum and minimum values of the register as the amplitude measurement. This value is subtracted from a set point amplitude, A_{sp} and passed as input to a PID controller with transfer function $H(s) = P + Ds + I/s$.

The parameters, P, I, and D of the PID controller were tuned manually, yielding

$$P = 0.1 \quad D = 0.0005 \quad I = 0.2.$$

The parameters were chosen to give a slow, stable response to changes in A_{sp} . For example, a step increase in A_{sp} from 0.25mm to 0.75mm takes approximately 10 seconds to stabilise, with virtually no overshoot. The output $A(t)$ of the controller is used as the amplitude for the actuator input, the output sinusoid passed to the actuator is $A(t) \cos 2\pi ft$.

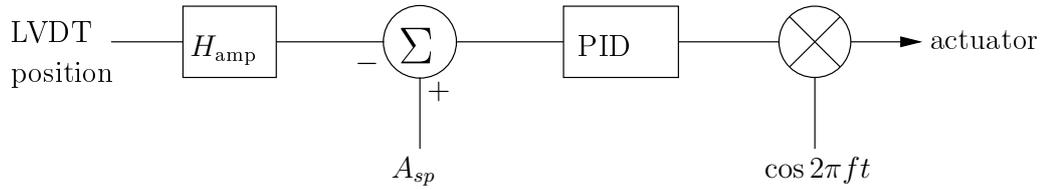
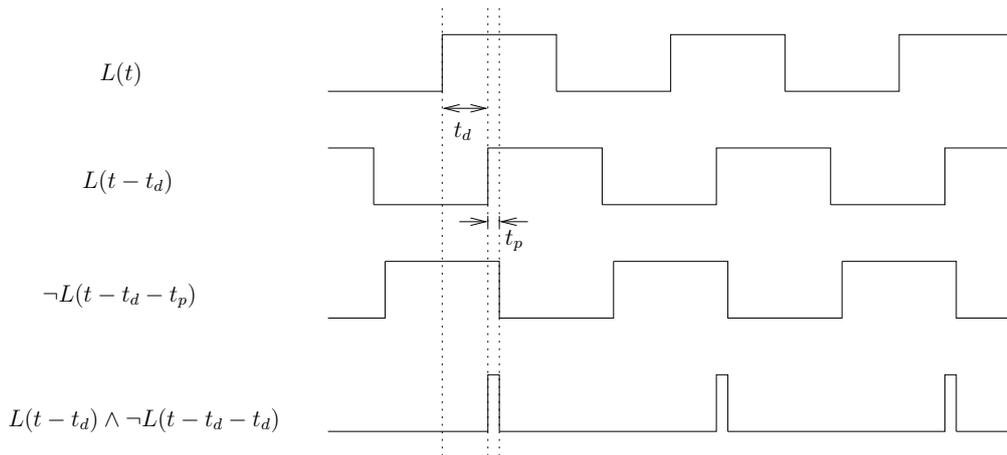


Figure 8.4: Actuator amplitude controller

8.3.1.2 Strobe signal generation

The strobe signal is synchronised to the LVDT signal as follows. The LVDT signal is first conditioned by passing it through a first order bandpass digital Butterworth filter with corner frequencies 10 and 400 Hz. The purpose of the filter is to remove the DC offset and high frequency noise in the signal. The filter is chosen to be first order to minimise the filtering delay. The filtered sinusoid is then converted to a square wave $L(t)$ by thresholding about 0.

Let the delay time be denoted t_d and the pulse on-time t_p . These times are computed from the frequency, duty cycle, and delay percentage. The pulse train is then generated as shown in Fig. 8.5 by computing $L(t - t_d) \wedge \neg L(t - t_d - t_p)$, where \wedge and \neg are Boolean AND and NOT, respectively. It should be noted that the construction of the signal depicted in Fig. 8.5 is only valid for duty cycles of up to 50%.

Figure 8.5: Computation of the strobe pulse train from thresholded LVDT signal $L(t)$

The strobe signal is synchronised with the LVDT, rather than the actuator input signal so that zero phase is aligned with the neutral position of the actuator plate.

8.3.2 Camera Server

The camera server software has two main functions. The first is interfacing with the cameras, and the second is coordinating dSpace and the cameras to run an experiment.

8.3.2.1 Camera initialisation and calibration procedure

The cameras are initialised to have the narrowest possible aperture setting (to provide the best depth-of-field), and an exposure time of 1 second at the minimum ISO setting (50). When the initialisation procedure is begun, these settings are sent to the cameras, and then the server waits until the user is ready with the calibration object. Once the calibration object is in place, the camera autofocuses, locks the focus, then takes an image of the calibration object. The user is then given the opportunity to retake the calibration images if required. Once the focus is locked, it remains locked until the camera server is terminated, guaranteeing the same camera settings on all images.

8.3.2.2 Experiment procedure

The sequence of events for taking a data set is shown in Fig. 8.6. The cameras are initialised and calibration is performed, a connection to the python server running in Control Desk (dSpace) is opened, and then the server starts listening for job requests from the client. The client sends a job request, which takes the form of a frequency, an amplitude, and a number of images required. The server sends the frequency and amplitude to dSpace, and then waits for 10 seconds for the system to stabilise. The server then tells the cameras to take an image. Once the image has been successfully taken, the server increments the phase delay in dSpace, and another set of pictures is taken. The process is complete when images at all of the required phase increments have been taken. Fig. 8.7 shows how the plot window of the GUI changes as the phase delay is increased

8.4 Results

A sample image taken from the imaging system is shown in Fig. 8.8. The phantom, although appearing stationary in the image, is being actuated at 100Hz at an amplitude of 0.75mm. The camera shutter was open for one second to capture the image. This image demonstrates that the strobe is correctly synchronised with the actuator signal.

Further results are deferred until Chapter 9, where the system is used to generate the images for a full 3D motion reconstruction experiment.

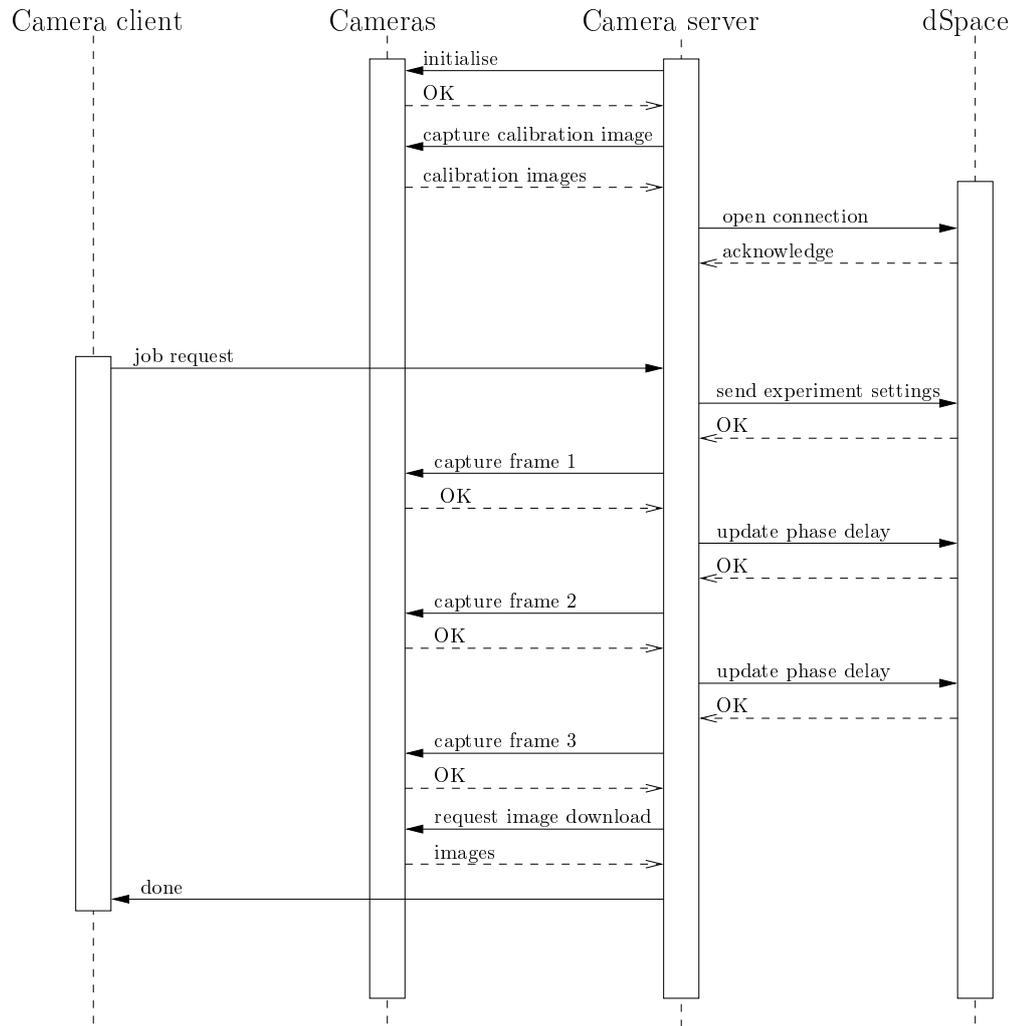


Figure 8.6: Sequence diagram for capturing a data set. Note, this illustrates a simple example where only 3 images are required for each camera.

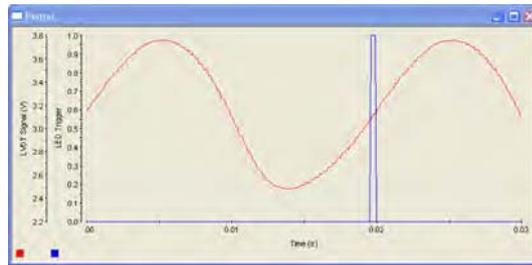
8.5 Labview

The image capture system described in this chapter has also been successfully implemented in Labview using a National Instruments CompactRIO and FPGA as the hardware platform. Essentially, this means that the dSpace, Control Desk, and GUI components in Fig. 8.1 are replaced with LabView equivalents. Therefore, the system architecture is not dependent on any particular control hardware implementation, as the Camera Server, which is responsible for experiment coordination can communicate with any network-enabled hardware system.

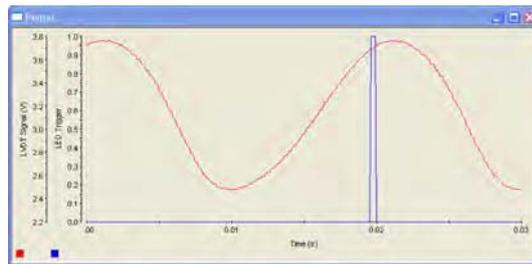
8.6 Summary

A hardware capture system has been developed that is capable of actuating silicone gel phantoms at a controlled frequency and amplitude, and using a strobe light coordinated with computer-controlled cameras to capture a set of images over a full phase of the motion of the phantom. The system has been designed using a modular approach, allowing system components to be upgraded or replaced as required. For example, the control software has been implemented and successfully used in Labview and dSpace.

The system has been used to successfully capture numerous data sets from silicone gel phantoms, and the results from these experiments will be discussed in Chapter 9.



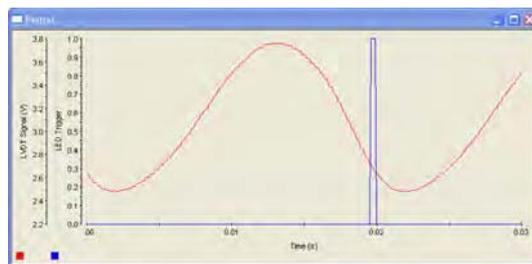
(a) delay = 0



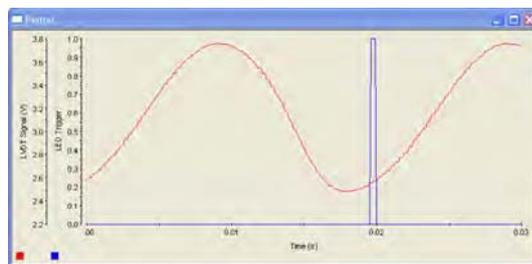
(b) delay = 0.2T



(c) delay = 0.4T



(d) delay = 0.6T



(e) delay = 0.8T

Figure 8.7: LVDT and strobe signals for different phase increments for a signal of period T . Note, the display is triggered by the strobe signal, and hence it appears that the strobe signal is stationary while the LVDT signal moves, when in reality it is the other way around



Figure 8.8: Still image of actuated gel phantom frozen by strobing. Note that the image is crisp and in focus

Chapter 9

CASE STUDY

9.1 Introduction

This chapter demonstrates the 3D motion reconstruction procedure in its entirety, from image capture through to motion reconstruction. The methodology developed throughout this thesis is presented step by step, showing intermediate results for this experiment. The experiments are performed on a gel phantom in the laboratory with four cameras.

Following this first case, two other experiments are presented. First, the number of cameras is increased to five to illustrate that five cameras are required for the DIET process on realistically sized phantoms. Second, results from three additional experiments measuring the motion of phantoms with different sized solid inclusions are presented. The different motion due to the inclusion is evident in each case. The experimental parameters were chosen to produce the maximum amount of motion on the gel phantom surface, as the object of the experiments is to test the 3D surface motion reconstruction capabilities of the algorithms presented in this thesis. The overall set of experiments are provided to create a complete proof of concept for the computer vision portions of the DIET system, as presented in this work.

9.2 Methodology

9.2.1 Gel phantom

A gel phantom was moulded out of an A-431 and LSR-05 silicone gel composite. The gel phantom is designed to have similar mechanical properties to human breast tissue [31,32]. The phantom is shown in Fig. 9.1, mounted above the actuator.

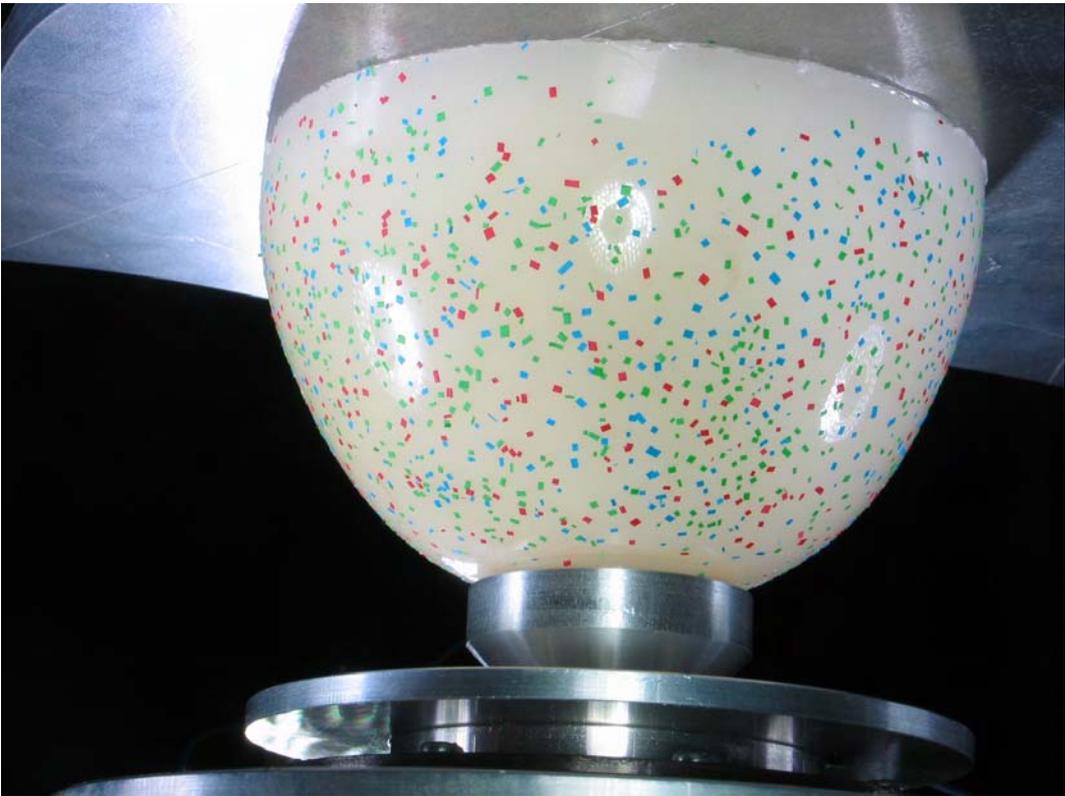


Figure 9.1: The gel phantom used. The phantom is suspended from an aluminium plate, and fits snugly in, but is not supported by, an aluminium interface screwed to the actuator plate

9.2.2 Camera calibration

The four cameras were calibrated using images of the black die (see Fig. 5.1 in Chapter 5). The cameras were approximately evenly spaced around the actuator oriented aiming up at the phantom. The cameras were at a radius of approximately 40cm from the centre of the actuator.

9.2.3 Fiducials

To get the best 3D reconstruction performance from the SEER algorithm of Chapter 7, fiducials were added to the phantom surface in three different colours: red, green and blue, as can be seen in Fig. 9.1. The fiducials are small pieces of paper (side length $< 1\text{mm}$). The phantom moulding process tends to statically charge the phantoms, which means that the fiducials stick very firmly to the surface, and are not dislodged by actuation of the phantom.

9.2.4 Image capture

The images are captured using the hardware system described in Chapter 8. It was found that the system settings that produced the maximum amount of motion without exceeding the capabilities of the actuator were an actuation frequency of 80Hz at an amplitude of 0.75mm. Each of the three phantoms was actuated at these settings, and a 20 frame sequence was captured from each camera, covering the entire phase at increments of 18° . The set of 20 images obtained from one camera is shown in Fig. 9.2

9.2.5 Feature extraction

The features were extracted using a simple thresholding procedure on the individual red, green, and blue colour channels of the colour images from the cameras. Three heuristics were devised for testing for red, green, or blue points. Let the set of image pixels be denoted \mathcal{I} and the red, green, and blue components of a pixel p be p_r, p_g, p_b respectively. The sets of red, green, and blue pixels R, G, B are defined:

$$\begin{aligned} R &= \{p \in \mathcal{I} \mid p_r > 80, p_r > 1.2p_g, p_r > 1.2p_b\} \\ G &= \{p \in \mathcal{I} \mid p_g > 80, p_g > p_b, p_g > 1.2p_r\} \\ B &= \{p \in \mathcal{I} \mid p_b > 80, p_b > p_g, p_b > 1.2p_g\} \end{aligned} \tag{9.1}$$

where pixel values range between 0 and 255 in each channel. These simple thresholds are able to accurately detect red, green, and blue regions in the images obtained under a reasonable range of lighting conditions. Pixels detected as red, green, and blue for the image in Fig. 9.1 are shown in Fig. 9.3a

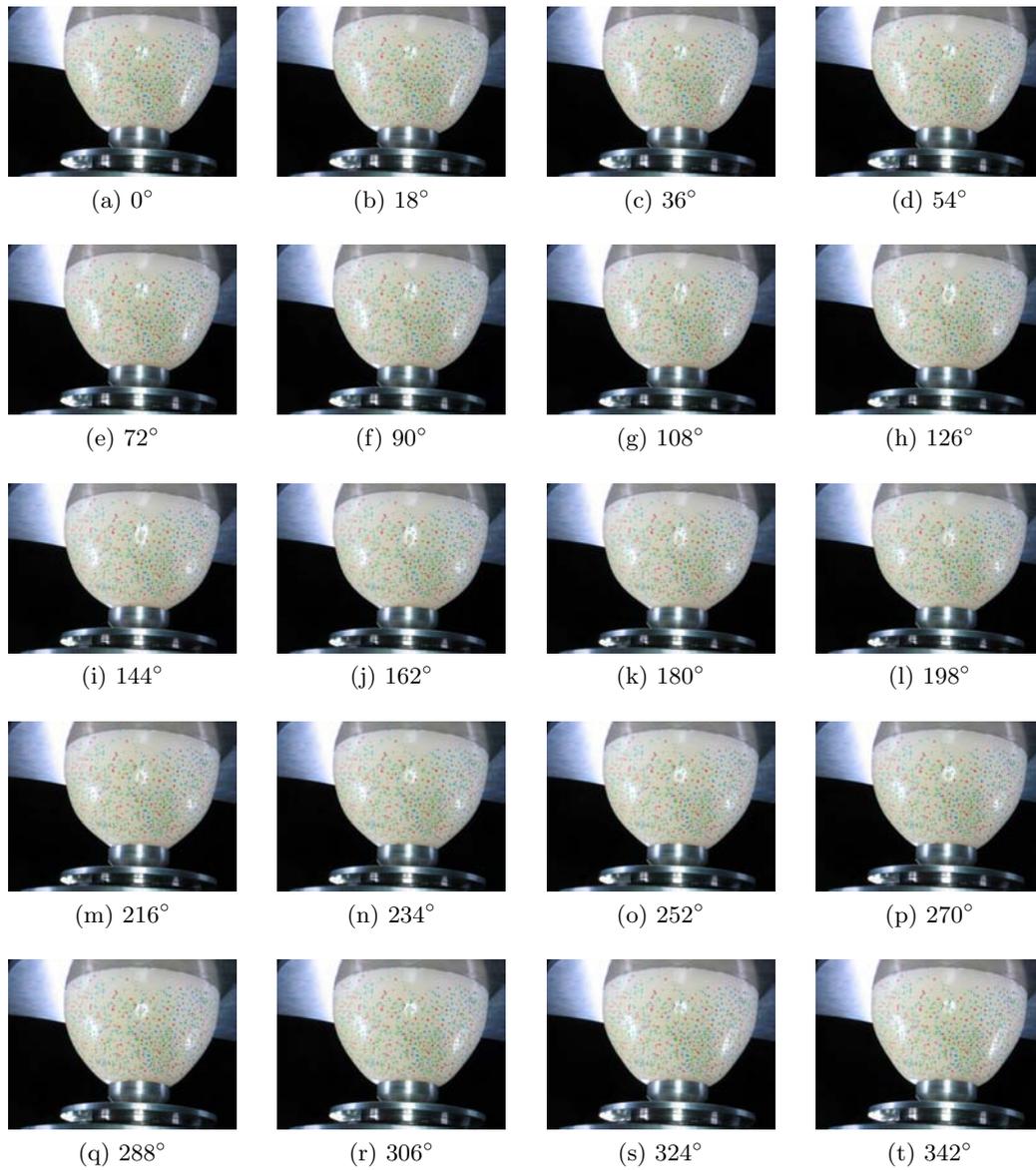
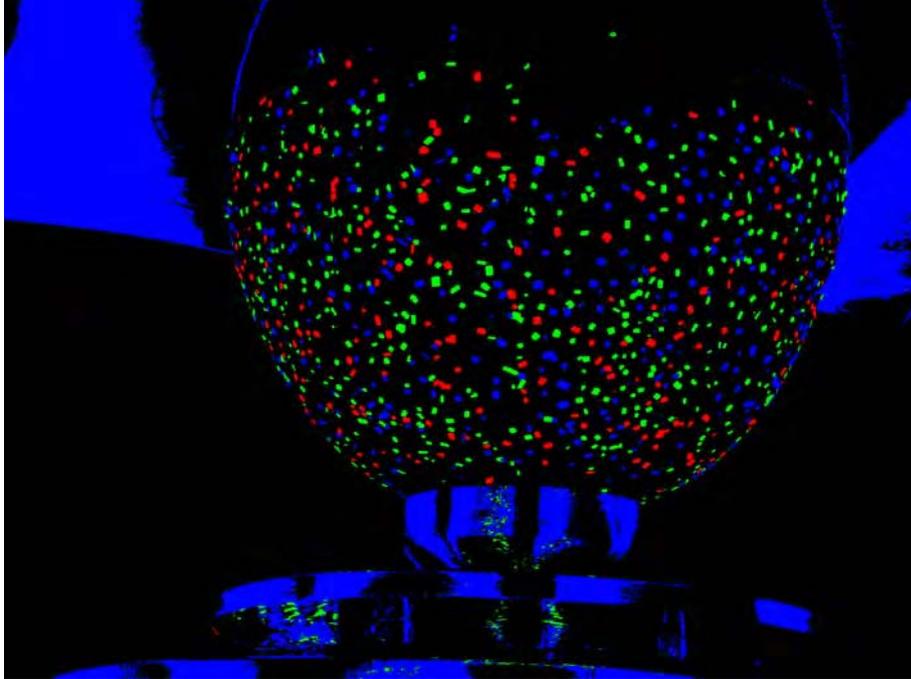
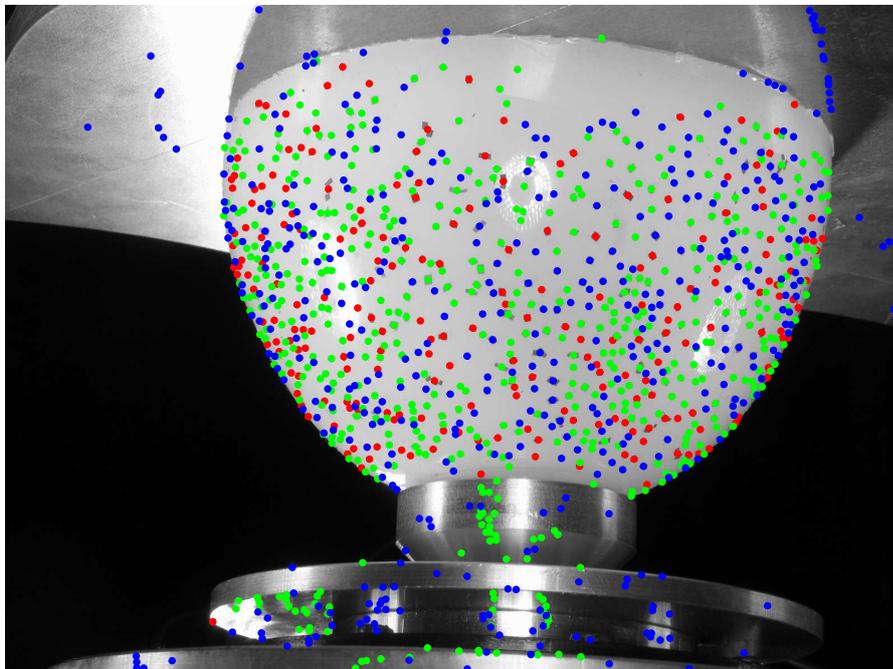


Figure 9.2: The 20 images obtained from one camera



(a) Pixels detected as red, green, and blue respectively in the image shown in Fig. 9.1. Note that the fact that extra regions are detected, such as the large blue regions, is not a problem, so long as the fiducials are clearly segmented from the background



(b) Centroids of the connected components

Figure 9.3: Feature extraction procedure

Once red, green, and blue regions are found, connected pixels of the same colour are found by performing a connected component analysis on the regions, and defining the image points to be the centroids of the connected components. The centroids for the image of Fig. 9.1 are depicted in Fig. 9.3b, where some erroneous centroid points are obvious.

9.2.5.1 Error analysis

There are two sources of error in the image point measurement procedure. The first source is the thresholding procedure (9.1). If pixels from an image point are missed out, or extra pixels are erroneously added, then error will be introduced into the centroid measurement. With the way that the colour thresholds are defined, it is much more likely that a coloured pixel will fail to be classified, than erroneously added, as the white background of the phantom will never pass any of the threshold conditions. The second source of error is the perspective distortion inherent in the image capture procedure. The centroid of the image region for a feature is *not* the image location of the centroid of the fiducial itself, although it is very close.

The effect of these sources of noise was investigated in a computer simulation. A hemisphere approximately the same size as the gel phantoms was covered with squares of side length 1mm, with uniformly randomly distributed centres and orientations. An image was generated from a camera placed at a similar position and orientation to the cameras used in the gel phantom experiments, with similar internal parameters. A black and white image of the fiducial regions was generated. All of the perimeter pixels of all of the fiducials in the image were pooled, and 50% of them were randomly discarded, simulating pixels that were not correctly identified as belonging to the region. The centroids of the resulting regions were computed, and compared with the image coordinates of the true centroid of the squares. The means μ_x, μ_y and standard deviations σ_x, σ_y were computed for the differences in the x and y coordinates. The results are shown in Fig. 9.4.

The errors are seen to be well-approximated by a Gaussian distribution in each coordinate, with mean 0 and standard deviation 0.2. This experiment is assumed to be representative of the image capture procedure, and gives a measure of the accuracy of the feature extraction stage.

9.2.6 Tracking

For the phantom experiments performed, the feature spacing was sufficiently sparse that the nearest neighbour tracking algorithm (Algorithm 6.1) was used to track the features. Features were tracked from frame 1 to frame 2, from 2 to 3, ..., from frame 19 to frame 20, and then from frame 20 back to frame 1. Any tracked feature that was not tracked back to itself after the 20 tracking steps was discarded. The tracked

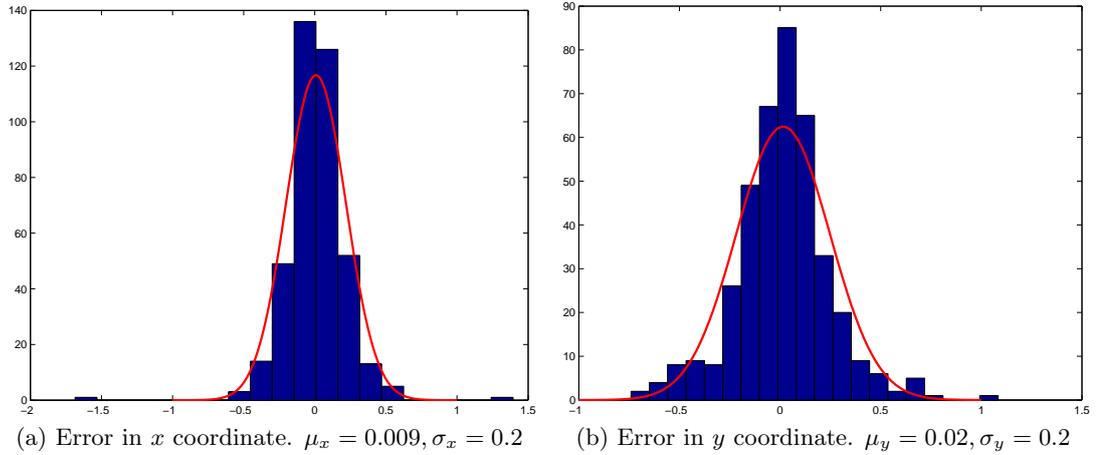


Figure 9.4: Errors in the measured centroids. The red lines show the Gaussian curves constructed from the computed means and standard deviations, showing that the error can be considered to be Gaussian

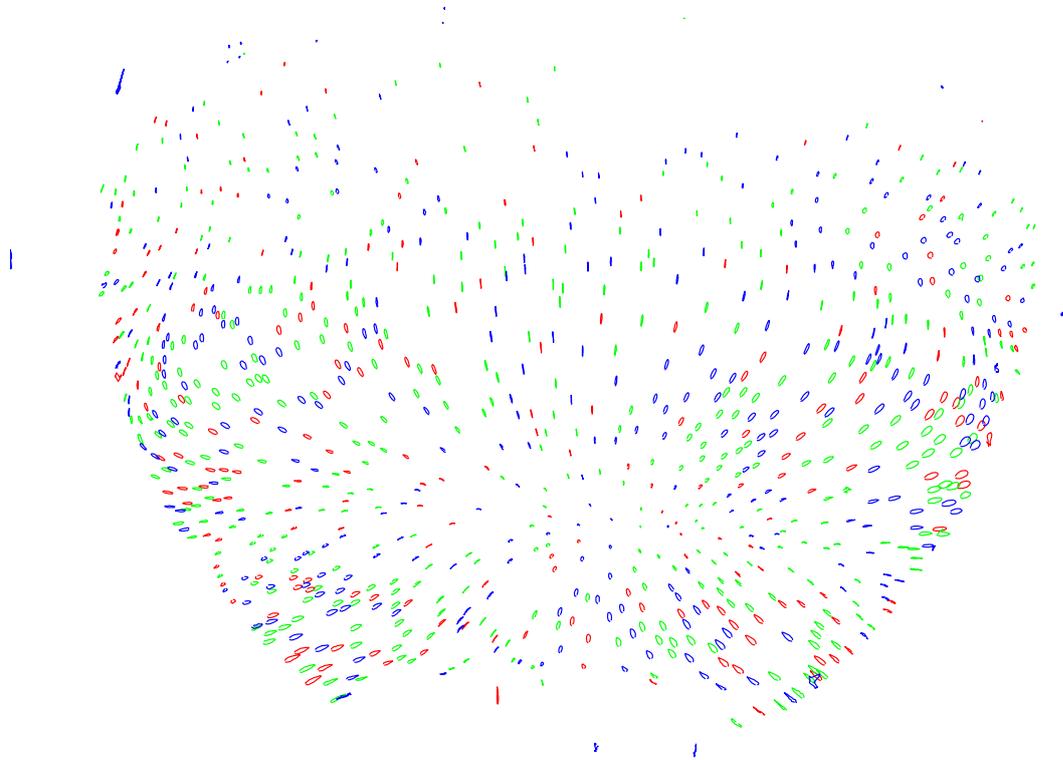
features from one camera are shown in Fig. 9.5, where the resulting elliptical shaped trajectories expected are evident.

9.2.7 3D surface reconstruction

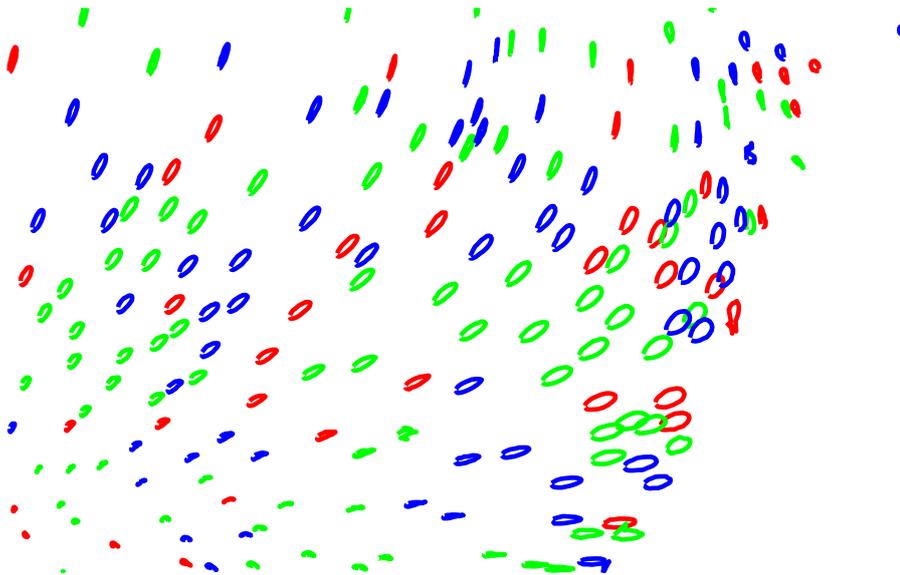
The SEER algorithm (Chapter 7) was used to reconstruct the features in 3D. To reduce image measurement noise, the centroids of each tracked feature were used as the input to SEER, rather than simply the coordinates in one frame. The surface was reconstructed using SEER, and the resulting feature correspondences were used to triangulate the matched features across all 20 frames.

An effective epipolar matching threshold of $\delta_{ep} = 2$ pixels was determined by Monte Carlo simulation. Image measurement error for the calibration images was modelled by a two-dimensional Gaussian distribution with standard deviation 0.25 in each coordinate. Image measurement error was assumed to be Gaussian with standard deviation of 0.5 pixels, which is higher than the value estimated earlier in this chapter. With this noise model added, camera resection was performed on the noisy coordinates and epipolar lines corresponding to nine keypoints in one image were drawn in the second image. This process was repeated 100 times. In each case, the epipolar lines corresponding to each point fell within a band of width around 4 pixels, so the threshold was chosen to be 2 pixels. Fig. 9.6 shows the details for this case.

Parameters used for SEER were: $n_B = 15, \kappa = 0.1, \rho_{est} = 8\text{pts}/\text{cm}^2$, and $d_{noise} = 0.2\text{mm}$. The reconstructed 3D points are shown in Fig. 9.7. The surface was meshed for visualisation purposes only, by converting to spherical polar coordinates and taking the Delaunay triangulation of the 2D coordinate system given by the azimuth and polar angles. Gaps can be seen in the sides of the reconstruction. These gaps are a



(a) All tracked features



(b) Close up

Figure 9.5: Tracked features from the image sequence in Fig. 9.2

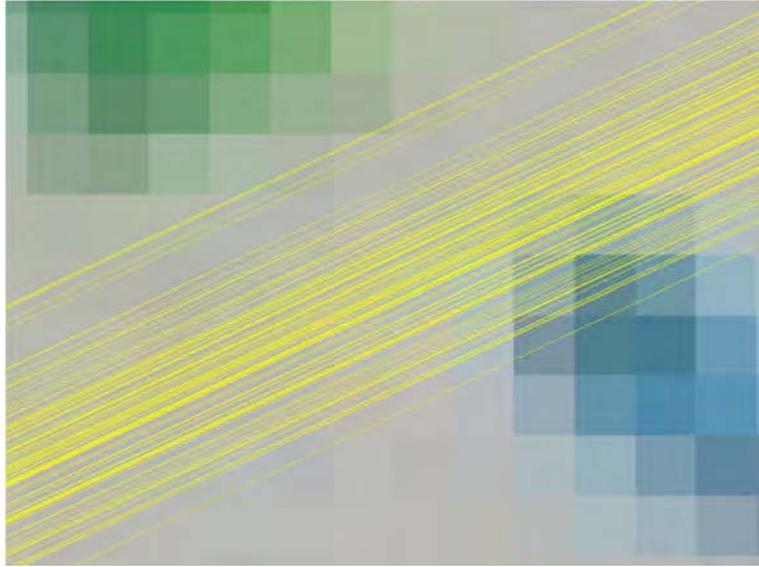


Figure 9.6: A band of epipolar lines generated by Monte Carlo simulation. The width of the band is approximately 4 pixels, so a matching threshold of 2 pixels is chosen

consequence of using only four cameras. The regions where no points are reconstructed are those regions that are only visible from one camera, indicating further cameras will be required for full coverage.

9.2.8 3D motion reconstruction

Once SEER has been used to construct the 3D surface, the image points used to construct a given 3D point, and hence their associated image trajectories, are in correspondence. The 3D trajectories of the points are computed by reconstructing each pair of points from corresponding image trajectories. The resulting 3D elliptical trajectories are shown in Fig. 9.8.

The mean image motion within a trajectory in each coordinate is approximately 6 pixels. The mean 3D motion within a trajectory in each coordinate is around 0.8mm. Assuming measurement noise of mean 0 and $\sigma = 0.2$ pixels, this allows an approximate estimate of the 3D error induced by the measurement noise to have mean 0 and standard deviation of $0.2 \cdot 0.8/6 \approx 0.02$ mm. The error in the measured motion resulting from a least squares fit of an ellipse to the trajectory will hence be lower, due to the averaging effect of having 20 points on the trajectory, and can be estimated to be approximately $0.02/\sqrt{20} \approx 0.005$ mm by the central limit theorem. Note that the error due to measurement noise is significantly lower than that induced by errors in the camera model, estimated to be of the order of 0.1mm, as described in §5. Note that the two sources of error are of a substantially different nature: The model error introduces a bias in a small region, whereas the measurement error source of random noise. The model error therefore does not contribute to the error in motion measurement.

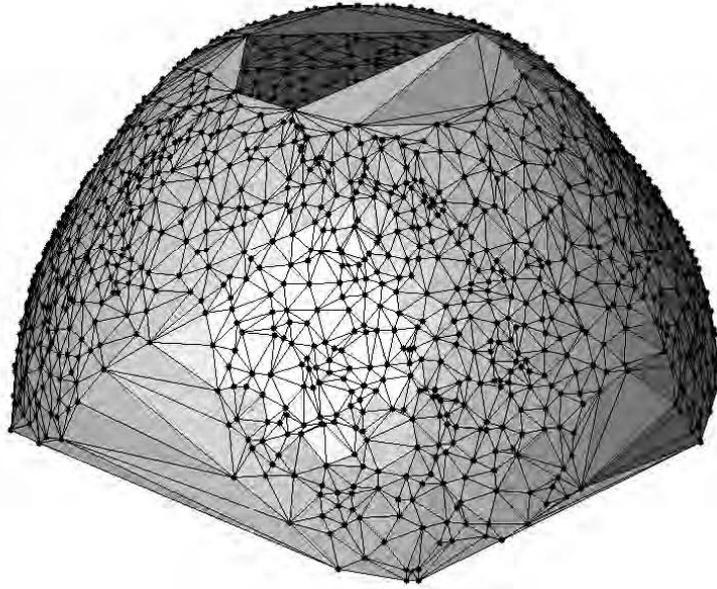


Figure 9.7: Reconstructed 3D points for the four-camera experiment. Note the gaps in the sides of the reconstruction due to insufficient cameras

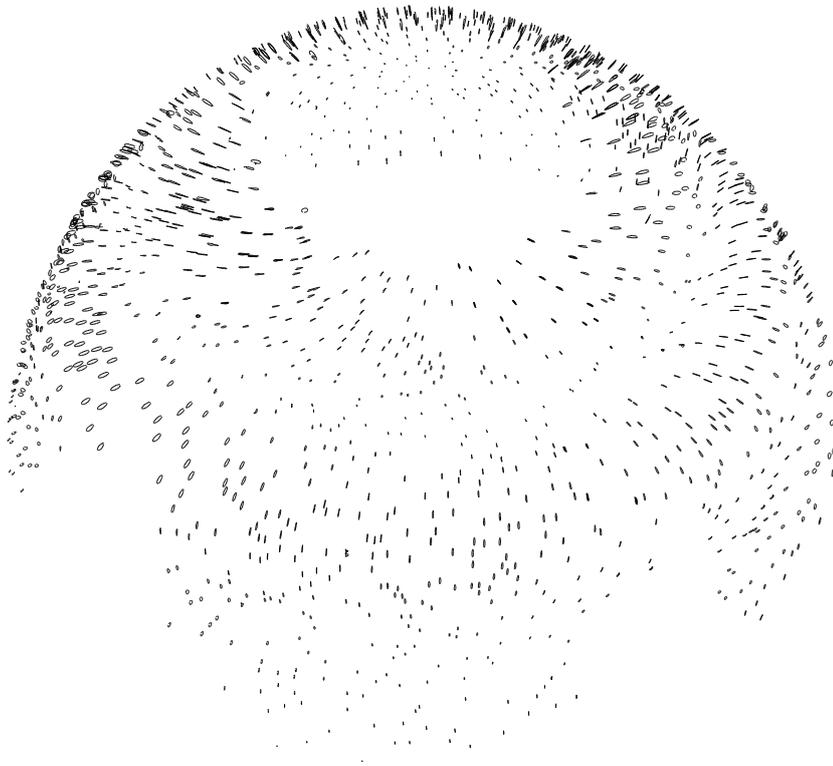


Figure 9.8: Reconstructed 3D motion of each point

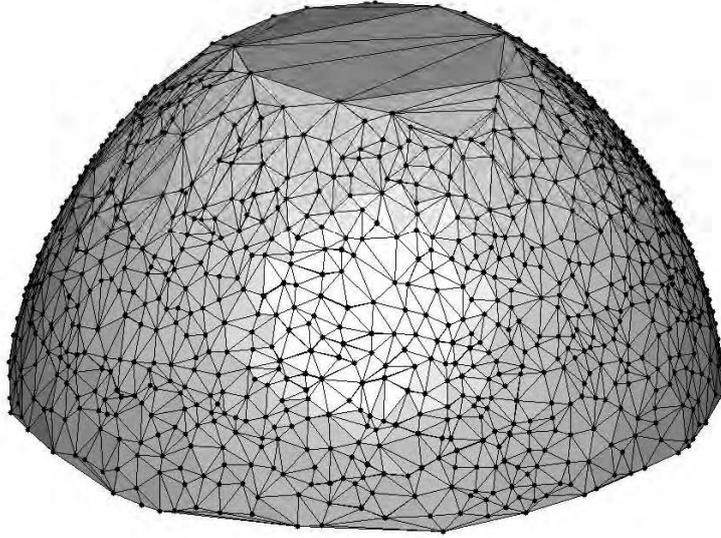


Figure 9.9: Reconstructed 3D points when five cameras are used. Note the gaps in the sides of the reconstruction from Fig. 9.7 are no longer present, as five cameras are sufficient

9.3 Experiments

9.3.1 Five cameras

The same phantom was used as in the case presented in §9.2. However, five cameras were used instead of four. The position and orientation of each camera can be seen in Fig. 5.11. Instead of using the black cube, the newer white calibration die was used, as presented in Chapter 5.

The surface reconstructed by SEER, using the same parameters as in §9.2 is shown in Fig. 9.9. It can be seen that if five cameras are used, the gaps in the reconstruction present when only four cameras are used are no longer present. This difference is evident in comparing Figs. 9.7 and 9.9. Five cameras is therefore the minimum number of cameras required for phantoms of this size, which are representative of the sizes a realistic DIET system might encounter.

9.3.2 Various sized inclusions

Four gel phantoms were prepared, containing respectively no inclusion, a small inclusion, a medium inclusion, and a large inclusion. The inclusions were made from silicone gel at approximately 4-5 times the stiffness of the rest of the phantom [31]. The large inclusion was 32mm in diameter, the medium inclusion was 22mm in diameter, and the small inclusion was approximately 15mm in diameter. The inclusions were placed with their centres offset 27mm from the central axis at a depth of 20mm from the phantom base (chest wall).

Four cameras were used, imaging a portion of the gel phantom, and the surface motion was reconstructed as in §9.2. Some of the reconstructed motion for the large inclusion phantom is depicted in Fig. 9.10. Note how in the close-up, panel (b), the orientation of the point trajectories appear to circle around a specific point. Contrast this behaviour with the close up of the motion of the phantom with no inclusion, panel (c) where the ellipses are aligned vertically. The presence of the large inclusion has a significant effect on the orientation of the trajectories of points on the surface. Fig. 9.10 thus shows that it may be possible to answer questions about whether or not there is an inclusion in the breast simply from the orientations of the 3D ellipses on the breast surface or similar analyses of the resulting motion.

9.3.2.1 One possible approach to detecting breast inhomogeneity from 3D ellipse characteristics

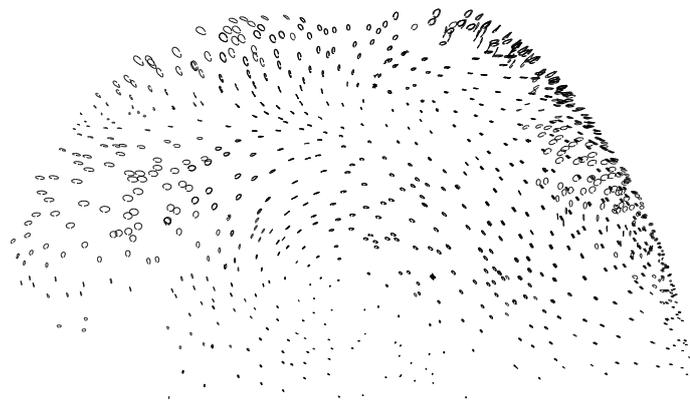
With reference to Fig. 9.10, it might be hypothesised that if there is no inclusion in the breast, then point trajectories are largely constrained to the plane defined by the direction of actuation and the surface normal. Out of plane motion may then be considered indicative of the presence of a hard inclusion, or a significant inhomogeneity in the breast. This out of plane motion component η was measured by taking the dot product of the normal to the hypothesised motion plane with the normal to a plane fitted to the ellipse coordinates.

For a given ellipse, $\eta = 1$ means that the ellipse motion is contained within the hypothesised plane, and $\eta = 0$ means that the ellipse motion plane is perpendicular to it. This out-of-plane motion component is computed for all measured ellipses for each phantom in §9.3.2. The results are shown in Fig. 9.11. Note that as the size of the inclusion increases, the amount (proportionally) of out of plane motion also increases.

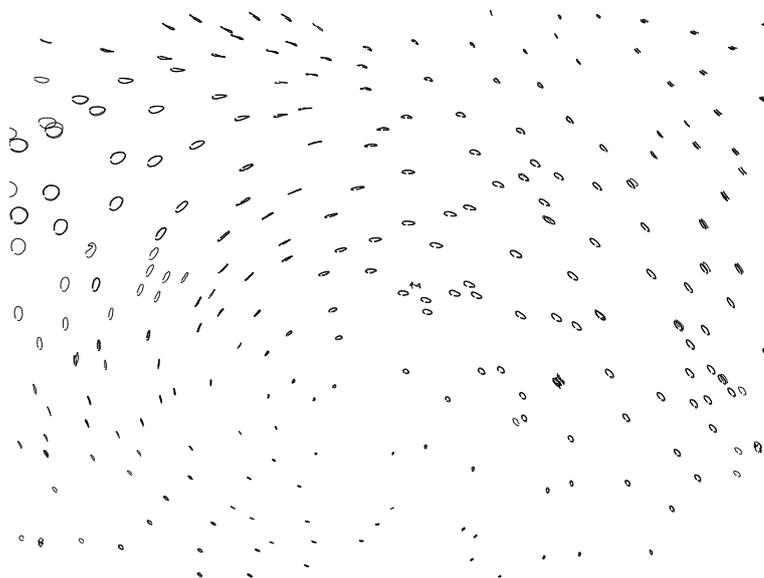
This analysis is one, initial non-rigorous example of how future investigations could proceed. For instance, while it may well be that η is a measure of non-homogeneity of the breast, it may equally well be that low values of η simply result from asymmetry in the breast. A more complex or surface-specific form of η might then be required. In any case, it is seen that adding an inclusion makes a qualitative difference to the 3D point motion on the breast surface.

9.4 Summary

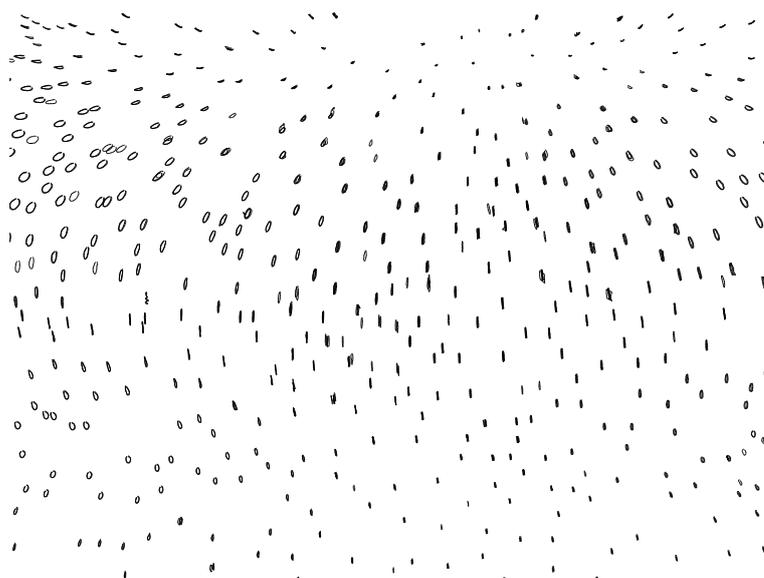
This chapter used a full phantom reconstruction experiment to describe how the various components of the 3D motion capture function together. The results form a proof-of-concept of the operation of the imaging system, as the gel phantoms are designed to be representative of what may be encountered in the real DIET system. Surface motion



(a) Motion of large inclusion phantom



(b) Close-up – large inclusion



(c) Close-up – no inclusion

Figure 9.10: Motion of the phantom with large inclusion

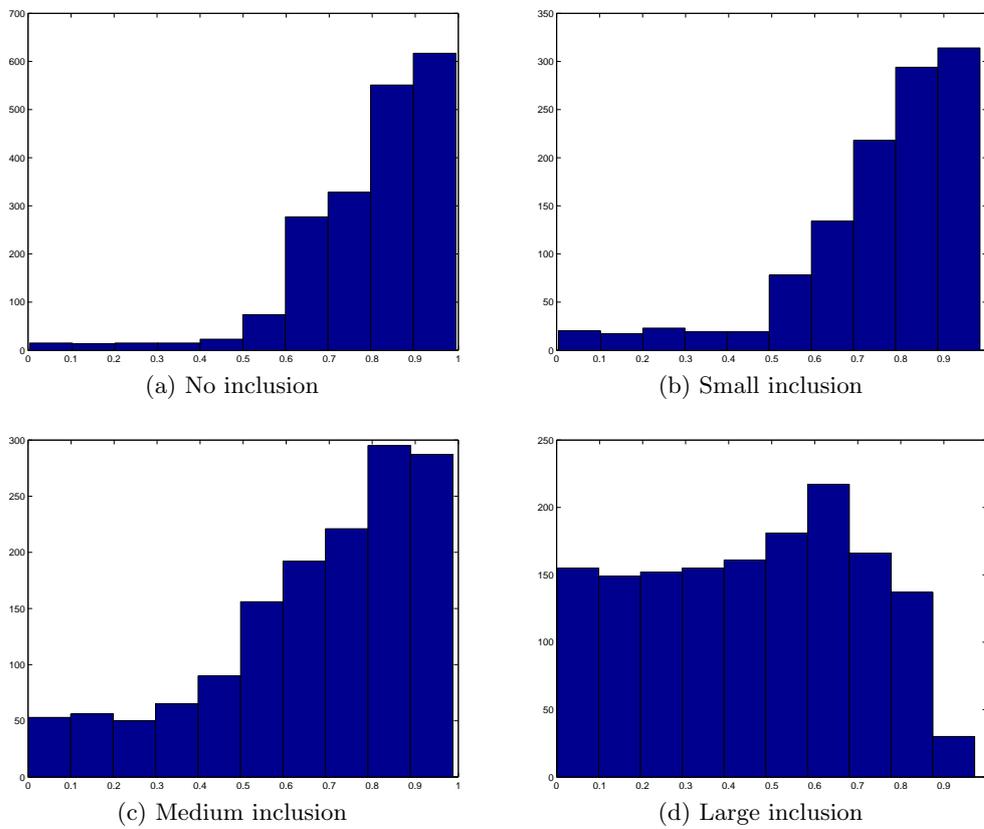


Figure 9.11: Measured out of plane component η for all 3D trajectories for phantoms with no inclusion, and small, medium, and large inclusions respectively

of approximately 1500 feature points was successfully obtained from the entire surface of the phantom.

The experiments verified the correct operation of the hardware system. Fluctuations in amplitude or irregularities in strobe timing would have resulted in fuzzy or blurry images. All the images from each camera were sharp and in focus, despite having an exposure time of 1.0 second.

The reconstructed elliptical point trajectories in 3D were clearly evident. These ellipses were used to illustrate some of the qualitative features of the surface motion in the presence of hard inclusions inside the gel phantom. The resulting analysis, while only exploratory, illustrated that there is a visible difference in the surface motion between a phantom with an inclusion and one without.

Chapter 10

CONCLUSIONS

The research presented in this thesis has developed the imaging component of the DIET system to a level where it is ready, without major modification, to be used in proof-of-concept and further clinical trials. The major objective of the research was to develop a system for effectively and very accurately capturing the 3D motion of a large number of points spread over the breast surface using a relatively minimal number of relatively low-cost cameras. This objective has been met, the image capture system is capable of capturing the motion of hundreds or even thousands of points to an estimated positional accuracy of approximately 0.2mm on the surface under actuation, and with motion measurement accuracy of the order of 0.01mm, or approximately 1% of the actuation amplitude.

The low-level hardware and software image capture system has been developed in a modular manner that allows for relatively easy future modification or substitution of any components. The system has also been implemented in LabView as well as dSpace. Hence, it provides a foundation for developing more extensive systems.

The main components of the image capture system are camera calibration, feature tracking, and 3D surface reconstruction. Camera calibration is critical to the integrity of the 3D reconstruction, as it provides the geometric framework of the image capture system. Any 3D reconstruction results are computed relative to the 3D space determined by the calibrated cameras. Feature tracking is necessary to determine how individual points move. Once a feature has been tracked, it needs only to be matched to a feature from a different camera once, as all remaining points on the matched trajectories will then automatically match as well. Finally, the 3D reconstruction stage estimates the 3D surface from amongst a point cloud of potential surface points. Once this point cloud has been found, this effectively means that correspondences have been found between points in pairs of images, and these correspondences enable the 3D reconstruction of the entire trajectory of each constructed 3D point.

A system for camera calibration has been developed that allows fast, accurate, fully

automatic calibration of an array of cameras from a single image of a fixed calibration cube from each camera. The calibration method allows for a wide range of possible camera viewpoints and is not restricted to one particular camera array geometry. The algorithm was designed to be general so that more sophisticated camera models can be incorporated as required in the future. More detailed calibration objects can also be used to provide greater accuracy. Thus, cameras with wide-angle lenses could be used to increase the field of view if the camera model was modified to correct for the appreciable amount of radial distortion present in images from wide-angle lenses.

Multiple algorithms for geometrically tracking random high-density point features between frames from one camera, have been developed. The density of points used in the full surface reconstruction experiments in this thesis was not sufficiently high to warrant using the invariant signature tracking approach. However, it will become useful if higher density point sets are considered, such as might be extracted from using the natural textures of human skin instead of applied fiducial markers.

The discrete signature approach, particularly for non-coloured points, is of potential interest in the more general field of pattern matching. The signatures provide a means of matching random point sets under non-rigid transformations that are able to be locally approximated by a transformation from a transformation group such as the Euclidean or similarity group. The idea was used in this thesis in an intra-image feature tracking context, so it was possible to enforce an upper bound on the magnitude of the motion between frames, thereby limiting the number of potential candidate matching points, and minimising the probability of mismatching points. In a more general setting, a robust outlier rejection procedure could be incorporated to reject mismatched points by requiring locally consistent motion between neighbouring points.

A 3D reconstruction algorithm, SEER, was designed for reconstruction of the breast surface from random point locations in pairs of images. SEER uses an alternative approach to conventional 3D reconstruction techniques. In this case, the search for point correspondences, and thus 3D points, is performed in the 3D space itself, rather than in the image frames. The advantage of this approach is that random features which exhibit a high degree of self-similarity can be used to perform the 3D reconstruction without requiring a difficult to obtain solution to the resulting correspondence problem. The other benefit of searching for the surface in 3D space, is that known properties of the surface, such as a bound on curvature and feature density, can be directly incorporated into the algorithm. SEER was designed for the DIET system, but the key principles are suitable for any application requiring smooth surface reconstruction from dense point features.

Laboratory experiments were performed on gel phantoms, designed to mimic clinical conditions as closely as possible. In these experiments, the motion of a large set of point features was successfully reconstructed in 3D, and emerging results in the DIET

project show that the motion data has been successfully used to identify and locate solid inclusions in the gel phantoms. The methods behind the elastographic reconstruction techniques are presented in the thesis by Peters [31].

In summary, the major requirement of this research was the full development of a system for 3D motion capture of an actuated breast. The key algorithmic components of the system were identified to be camera calibration, feature tracking, and 3D reconstruction. Additionally, to successfully capture laboratory data sets for proof-of-concept case studies, a low level actuation control and image capture system was required. All of these requirements were successfully accomplished, together with some more widely applicable algorithms and techniques in the wider pattern recognition and computer vision fields.

Chapter 11

FUTURE WORK

11.1 DIET

The next major aim for the DIET project as a whole, is to obtain data from clinical experiments on human patients. The image-capture hardware system will require some modification to make it suitable for use in a medical context. An example of what the hardware might look like is depicted in Fig. 11.1. A clinically-suitable method is required for randomly applying the coloured fiducials to the breast surface.

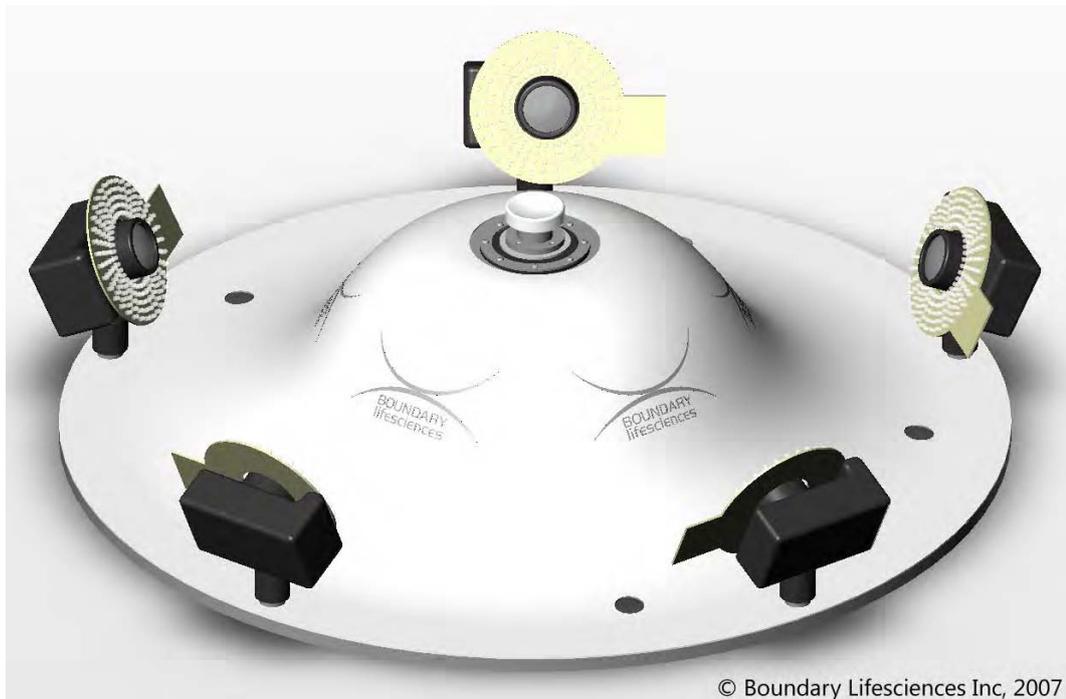


Figure 11.1: Possible clinical experimental setup

Once clinical data has been successfully obtained, a highly desirable outcome is being able to replace the coloured fiducial system with features extracted directly from

the breast surface. Some preliminary research has been performed to this end [19], with natural skin features being tracked successfully by a variant of the NCC (normalised cross-correlation) algorithm.

Another important task for clinical viability of the image capture system is making the various algorithms robust. The current feature extraction, tracking, and 3D reconstruction algorithms all require a number of tuned parameters, which may differ between patients. Therefore, a method of automatically estimating suitable parameters is required, so that the system operator is not required to have a highly technical knowledge of the operation of the algorithms.

11.2 Camera calibration

Presently, the design of purpose-built cameras for the DIET system is being considered. Depending on the choice of lens for these cameras, the camera model used in camera calibration may need to be updated to incorporate radial distortion. Usage of this model will require the development of a calibration cube with more feature points on it to account for the estimation of the additional model parameters.

11.3 Signature tracking

The approach of matching random point patterns by locally computed invariant signatures has potential applications in numerous contexts. The signature definition can easily be extended to other transformation groups, however with less restrictive transformation groups, noise has a tendency to be amplified in the normalisation procedure, introducing large errors in the signatures. By incorporating a robust outlier-rejection procedure which forces correspondences to satisfy a smoothly varying global transformation, it may be possible to extend the signatures successfully to groups such as the projective or affine group. If a successful projective-invariant signature was developed, this could be used to solve the correspondence problem for two views of a random point set on a smooth surface from two separate cameras, as locally a smooth surface can be approximated by a plane, and a world plane induces a projective transformation between images of the plane from two cameras.

11.4 SEER

The SEER algorithm has been shown to work successfully for the DIET project at a variety of feature point densities. SEER has potential for application in other 3D surface construction applications, however there are a few areas that would first need to be addressed. The first is relaxing the requirement of uniform feature density throughout the

surface, as for large flat regions, few features are required to accurately represent the surface, whereas for regions of high curvature, a higher density of features is required. This could be achieved by estimating the surface density based on the relative densities in the image. Design of more sophisticated local surface fitting routines and neighbourhood definitions could allow recognition of non-smooth features (such as edges), which would significantly widen the class of objects that could be successfully reconstructed.

REFERENCES

- [1] C. C. Boring, Squires, and T. Tong. Cancer statistics. *CA Cancer J. Clin.*, 44:7–26, 1994.
- [2] E. Calabi, P.J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker. Differential and Numerically Invariant Signature Curves Applied to Object Recognition. *International Journal of Computer Vision*, 26(2):107–135, 1998.
- [3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [4] JG Chase, EEW Van Houten, L. Ray, D. Bates, JP Henderson, CM Ewing, and C. Berg. Digital Image-Based Elasto-Tomography for Soft Tissue Imaging. *Osaka, Japan: First Asian Pacific Conference on Biomechanics (APBiomech 2004)*, pages 25–28, 2004.
- [5] P. Chaturvedi, M. Insana, and T. Hall. Acoustic and elastic imaging to model disease induced changes in soft tissue structure. In *Information Processing in Medical Imaging*, Proc. IMPI, 1997.
- [6] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.
- [7] SR Deans. Hough Transform from the Radon Transform. *IEEE TRANS. PATTERN ANALY. AND MACH. INTELLIG.*, 3(2):185–188, 1981.
- [8] John D’Errico. Matlab Central File Exchange — Surface Fitting using gridfit, 2006. [Online; retrieved 11-October-2006].
- [9] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.
- [10] M. Fels and P.J. Olver. Moving coframes. I. A practical algorithm. *Acta Appl. Math.*, 51(2):161–213, 1998.

- [11] M. Fels and P.J. Olver. Moving Coframes: II. Regularization and Theoretical Foundations. *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, 55(2):127–208, 1999.
- [12] M.A. Fischler and R.C. Bolles. 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.
- [13] W. Gander, G.H. Golub, and R. Strebler. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, 1994.
- [14] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
- [15] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, 15:50, 1988.
- [16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
- [17] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112, 1997.
- [18] C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3d object modelling. *CVIU*, 96(3):367–392, 2004.
- [19] AJH Hii, CE Hann, JG Chase, and EEW Van Houten. Fast normalized cross correlation for motion tracking using basis functions. *Computer Methods and Programs in Biomedicine*, 82(2):144–156, 2006.
- [20] P. D. Kovesi. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: <<http://www.csse.uwa.edu.au/~pk/research/matlabfns/>>.
- [21] T. M. Krouskup, T. Wheeler, F. Kallel, B. S. Garra, and T. Hall. Elastic moduli of breast and prostate tissues under compression. *Ultrasonic Imaging*, 20:260–274, 1998.
- [22] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.
- [23] M. Li, M. Magnor, and H.P. Seidel. Hardware-Accelerated Visual Hull Reconstruction and Rendering. *Graphics Interface Proceedings 2003: Canadian Human-Computer Communications Society*, 2003.

- [24] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [25] Y. Ma et al. *An invitation to 3-D vision*. Springer, 2004.
- [26] The Mathworks. Radon transform documentation, 2008. [Online; retrieved 22-January-2008].
- [27] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25-27, 2001*, 2001.
- [28] Cancer Society of New Zealand. Causes of cancer death in New Zealand, 2003.
- [29] P.J. Olver. Joint Invariant Signatures. *Foundations of Computational Mathematics*, 1(1):3–68, 2001.
- [30] D Pedoe. *Geometry: A Comprehensive Course*. Courier Dover Publications, 1988.
- [31] A. Peters. *Digital Image Elasto-Tomography: Mechanical Property Reconstruction from Surface Measured Displacement Data*. PhD thesis, University of Canterbury, 2007.
- [32] A. Peters, S. Wortmann, R. Elliott, M. Staiger, J.G. Chase, and E.E.W. Van Houten. Digital Image-based Elasto-Tomography: First experiments in surface based mechanical property estimation of gelatine phantoms. *JSME International Journal*, 48:562–569, 2005.
- [33] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. *CVPR*, II:882–827, 2005.
- [34] A. Sarvazyan. Mechanical imaging: a new technology for medical diagnostics. *Int. J. Med. Inf.*, 49(2):195–216, 1998.
- [35] S.M. Smith and J.M. Brady. SUSAN – A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.
- [36] I.E. Sobel. *Camera models and machine perception*. PhD thesis, Stanford University, 1970.
- [37] M. Tilanus-Linthorst, L. Verhoog, I. M. Obdeijn, K. Bartels, M. Menke-Pluymers, A. Eggermont, J. Klijn, H. Meijers-Heijboer, T. van der Kwast, and C. Brekelmans. A brca1/2 mutation, high breast density and prominent pushing margins of a tumor independently contribute to a frequent false-negative mammography. *Int. J. Cancer*, 102(1):91–95, 2002.

- [38] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998.
- [39] R. Y. Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *Radiometry*, 1992.
- [40] J. B. Weaver, E. E. W. Van Houten, M. I. Miga, F. E. Kennedy, and K. D. Paulsen. Magnetic resonance elastography using 3d gradient echo measurements of steady-state motion. *Medical Physics*, 28(8):1620–1628, 2001.
- [41] J. Weng, P. Cohen, and M. Herniou. Camera Calibration with Distortion Models and Accuracy Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.
- [42] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.
- [43] Y. Zheng and D. Doermann. Robust Point Matching for Nonrigid Shapes by Preserving Local Neighborhood Structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):643–649, 2006.