Reconstruction of Extended Environments from Image Sequences

by

Stuart Butterfield

Submitted in accordance with the requirements

for the degree of Doctor of Philosophy

The University of Leeds

School of Computer Studies

May 1997

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

Stuart Butterfield

School of Computer Studies
University of Leeds

Doctor of Philosophy
May 1997

Reconstruction of Extended Environments from Image Sequences

Abstract

The automatic recovery of the three-dimensional structure of a scene from a sequence of two-dimensional images has been the subject of considerable research in the field of machine vision, with applications as wide-ranging as object recognition, virtual reality and robot navigation. Traditional attempts to solve this *structure from motion* (SFM) problem rely on calibrated cameras and involve the detection and tracking of features through successive images in the sequence.

When considering long image sequences, taken with an ordinary hand-held video camera, the problem is significantly harder, since both camera calibration parameters and matched feature information are difficult to obtain accurately. An additional complication is that small errors in the recovered structure will accumulate over long sequences, possibly resulting in a reconstruction which is internally inconsistent. To date, there has been no discussion in the SFM literature of attempts to tackle this important issue.

Recently, a number of different techniques have been developed for scene reconstruction using uncalibrated cameras. In such cases the recovered structure is correct up to a projective transformation of the real structure. In this thesis, an original, incremental reconstruction system is described, based on this uncalibrated approach. A novel implementation for computing the *fundamental matrix* from a pair of images is presented, from which a projective reconstruction is obtained. For the first image pair in the sequence, a small number of ground truth points are used to upgrade from projective to Euclidean structure. This structure is propagated through successive frames to obtain a complete Euclidean reconstruction for the entire scene. The inconsistency problem is addressed by attempting to detect when previously viewed sections of the scene are re-encountered. A solution method using the geometric hashing model-based object recognition paradigm is proposed.

Acknowledgements

First of all, I would like to thank Professor David Hogg for his encouragement, guidance and patience throughout the duration of my research.

Too numerous to mention are all those staff and students in the School of Computer Studies, who have helped make my time here so enjoyable. Particular, thanks must go to my colleagues in the Vision Group, both past and present, who created such a friendly atmosphere to work, and play, in. From demonstration classes to doughnuts, from research seminars to football, it's been a pleasure.

A special thank-you to Andy Bulpitt, proof-reader, agony uncle and a great friend. I know I wouldn't have reached this point without him.

My eternal gratitude goes to my parents and sister for their incredible support, especially during this last year, and thanks to Sarah, for everything, but especially for putting up with me for so long.

I gratefully acknowledge the financial support of the University of Leeds, for the first three years of this research, and thank the School of Computer Studies for keeping me gainfully employed whilst writing up.

Finally, for her unwavering belief, to Elizabeth, the usual.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work described in this thesis is motivated by a seemingly simple task - to determine the structure of a scene as viewed in a segment of video footage, typically obtained using an ordinary hand-held video camera. Our particular concern is with ‘extended environments’, for which a small part of the scene is visible in each frame. Reconstruction of such environments introduces associated problems in ensuring the internal consistency of the recovered structure. In particular, structures that are seen more than once during the image sequence must be correctly identified with each other.

In terms of machine vision, the task is a combination of the fundamental problems of *structure from motion* (SFM) and *structure matching*, which have, and continue to be, extensively researched. The aim of this work is to bring together ideas and techniques from these two areas in an attempt to solve the overall task at hand. A successful system would have many, varied uses, for example in constructing a virtual reality ‘walk-through’ of a real building, navigating a moving robot, automatic acquisition of CAD models etc. Ideally, it should not be dependent on the type of camera motion, or the nature of the scene being viewed. Sample images from the kinds of sequences such a system might be expected to deal with are shown in figure 1.1.

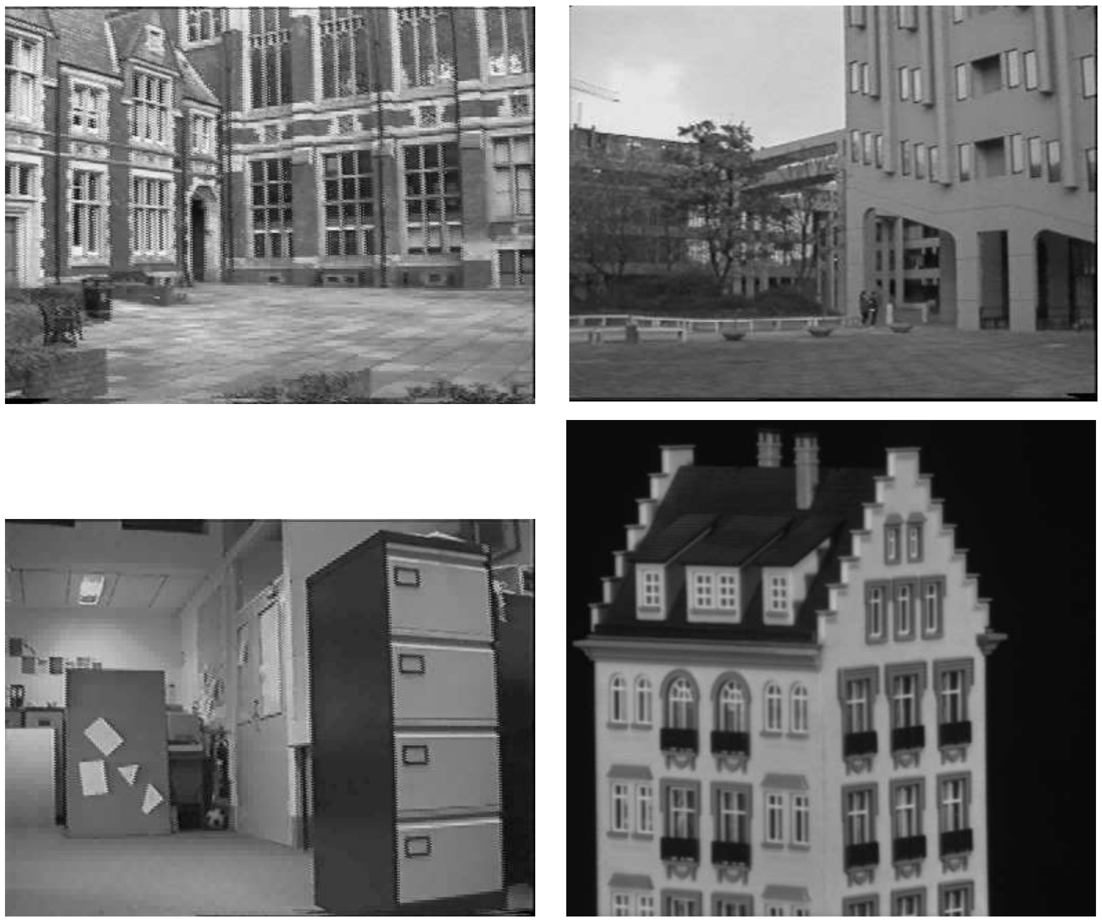


Figure 1.1: Sample images.

1.1 Approach Taken

The choice of approach is dependent on the nature of the input image sequence. In our case the large disparity between successive images called for the use of feature-based SFM, rather than optical flow [1]. A manual corner detection and matching system is used, to avoid the additional difficulties caused by poor localisation and false matches inherent in automatic methods.

Even if all the images in the sequence are taken with the same camera (which is likely, but not certain), there is no guarantee that the internal camera parameters do not change during the course of the sequence, for example due to zooming. Thus it is necessary to employ an uncalibrated reconstruction technique. Note that even camera *self-calibration* [13] is not possible, since this depends on unchanging intrinsic parameters. The procedure is based around the calculation of the fundamental matrix, which embodies the epipolar geometry of a pair of images. The calculation is performed via a novel combination of the well-known 8-point algorithm [39], a recently developed normalisation technique [26] and the RANSAC parameter estimation paradigm [15].

The fundamental matrix is factorised to obtain a representation for the camera matrices, and the structure of the scene in the two images is recovered by back-projection. The reconstruction so obtained is only correct up to a projective transform of the real structure. Knowledge of a small number of ground truth points in the first image pair is used to compute a projectivity, which transforms the projective structure to Euclidean. For subsequent image pairs, the projectivity calculation is performed using previously estimated Euclidean structure instead of ground truth. Thus, as the sequence is processed, a complete reconstruction of the scene is incrementally acquired.

The second major part of this work is motivated by the acceptance of the fact that the reconstruction system will not produce perfect results, and as the sequence is processed, errors in the recovered structure will accumulate. If part of the scene is re-encountered, it will have two Euclidean structure estimates, at some displacement and orientation in the world coordinate system. In order to be able to update the structure so that it is internally consistent, it is necessary to obtain the mapping between the points in the two structure estimates. This, in turn, depends upon the ability of the system to recognise when structure that has been seen previously has come back into view.

These two problems are addressed by incorporating a model-based object recognition system into the reconstruction process. A variation of the standard geometric hashing algorithm is employed, whereby model acquisition and recognition are performed concurrently, the usual off-line preprocessing step being omitted. At each step of the reconstruction process, a new segment of structure is recovered. This is combined with other recently acquired structure to obtain a local *structure patch*. The hash table is updated using Euclidean invariants computed from this set of 3D points. Thus, recognition should occur if any of these patches of structure are re-encountered later in the sequence. In this case the information stored in the hash table permits the mapping between old and new scene structure to be immediately obtained.

1.2 Overview of the Thesis

The remainder of the thesis is organised as follows:

Chapter 2: A review of the relevant background material and related research.

Chapter 3: A description of a novel method used to estimate the fundamental matrix, based on a normalised version of the 8-point algorithm.

Chapter 4: A description of an original incremental reconstruction system for recovering the Euclidean structure of a scene from a long image sequence.

Chapter 5: A description of a version of the geometric hashing algorithm which has been incorporated into the reconstruction system in an attempt to allow previously encountered scene structure to be recognised.

Chapter 6: Conclusions and a discussion of future work.

Appendix A: An overview of some of the essential principles of projective geometry.

Appendix B: A detailed description of the RANSAC parameter estimation paradigm.

Appendix C: The quadrangle image sequence, which is used in experiments throughout this work.

1.3 Notation

We will use boldface letters to denote vectors and matrices e.g. $\mathbf{M}$. The notation M_{ij} denotes the element at the i'th row and j'th column of matrix $\mathbf{M}$. Transposition of vectors and matrices is indicated by ^{T}, e.g. $\mathbf{M}^T$, the inverse of a matrix $\mathbf{M}$ by $\mathbf{M}^{-1}$ and the inverse transpose by $\mathbf{M}^{-T}$. The determinant of a matrix is denoted $det(\mathbf{M})$. The cross product of two 3D vectors, $\mathbf{x}$ and $\mathbf{y}$ is represented by $\mathbf{x}\times\mathbf{y}$. Given some vector $\mathbf{t} = (t_x, t_y, t_z)$, it is useful to consider the anti-symmetric matrix

$$[\mathbf{t}]_\times = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{1.1}$$

This is the matrix representation of the cross product. For any vectors $\mathbf{s}$ and $\mathbf{t}$,

$$\mathbf{s}^T[\mathbf{t}]_\times = \mathbf{s}\times\mathbf{t} \tag{1.2}$$

and

$$[\mathbf{t}]_\times\mathbf{s} = \mathbf{t}\times\mathbf{s} \tag{1.3}$$

On a related note, given the square matrix $\mathbf{M}$, we use the notation $\mathbf{M}^*$ to represent the matrix of cofactors of $\mathbf{M}$, that is, the matrix defined by $M^*_{ij} = (-1)^{i+j}\det(\mathbf{M}^{(ij)})$ where $\mathbf{M}^{(ij)}$ is the matrix derived from $\mathbf{M}$ by removing the i'th row and j'th column. If $\mathbf{M}$ is non-singular then $\mathbf{M}^* = \det(\mathbf{M}).(\mathbf{M}^{-T})$. In other words $\mathbf{M}^* \approx \mathbf{M}^{-T}$, where $\approx$ indicates equality up to a scale factor. If $\mathbf{a}$ and $\mathbf{b}$ are 3$\times$1 vectors and $\mathbf{M}$ a 3$\times$3 matrix then $\mathbf{Ma}\times\mathbf{Mb}\approx\mathbf{M}^*(\mathbf{a}\times\mathbf{b})$.

Uppercase letters will be used to denote 3D points and lines, whereas lowercase letters indicate 2D features. We differentiate between the geometric objects themselves and their representations. For example, a point in the image plane would be denoted by p, whereas its coordinate vector would be $\mathbf{p}$. The line between two points P_1 and P_2 is represented by $\langle P_1, P_2\rangle$.

Chapter 2

Background

2.1 Introduction

The recovery of the three-dimensional information lost when projecting a scene onto an image plane is a central problem in machine vision. At least two images are required, either taken by a pair of cameras in a stereo configuration, or a single moving camera. Reconstruction via the latter of these is termed *structure from motion* (SFM). An early structure from motion theorem, due to Ullman [66], states that:

> Given three distinct orthographic projections of four non-coplanar points in a rigid configuration, the structure and motion compatible with the three views are uniquely determined, up to a reflection about the image plane.

Over the years there have been innumerable structure from motion algorithms described in the vision literature. Due to the assumption of orthographic projection[1], which is not an accurate model of the real image formation process, a general purpose structure recovery system based on this theorem is impractical. However, it does serve to illustrate a number of the ways in which structure from motion algorithms can be characterised.

[1]Where rays are projected from an object point along a direction parallel to the camera's optical axis, so that they strike the image plane orthogonally.

First of all, there is the question of how many projections (images) are required for the algorithm to function. Some systems work with just two or three images, others have these as a minimum number for structure recovery, but are able to deal with long sequences of images. With image sequences, there is the additional consideration of whether the images are actually processed sequentially or as a batch. The number and type of image features used are also varied.

Possibly the two most important characteristics are the projection model and whether or not the camera is calibrated. The most general method of projection from 3D space onto a 2D image is *perspective projection* (see Appendix A). Other methods approximate this under certain imaging conditions, for example *scaled orthographic* (weak perspective), *paraperspective* [48] and *affine* [47]. A description of the most appropriate camera model for a given situation is provided in [69], along with a method for switching models *during* processing to ensure the best model is used at all times. In our case, we make no assumptions about the nature of the scenes being viewed and so use the full perspective model.

If camera calibration[2] is known, scene structure can be recovered up to a scale factor [11]. However, the calibration process usually relies upon accurately measuring calibration objects and is very sensitive to errors [64]. Secondly, calibration has to be recomputed if the internal parameters change, for example, the camera zooms. These difficulties have led to the development of the projective approach to reconstruction, also known as *uncalibrated stereo* [52]. This method models the geometric relationship between the cameras, matched points and corresponding 3D positions, which is encapsulated in the *fundamental matrix*. The non-metric nature of this projective approach means there is no dependency on camera parameters, but recovered structure differs from the real structure by a projective transformation.

The acquisition of the fundamental matrix and a projective reconstruction technique form a significant part of this work. These are fully detailed in Chapters 3 and 4. We continue in this chapter with a more general background description of the two subjects. This is followed by a review of current research on reconstruction from image sequences, most relevant to this thesis. The chapter concludes with an overview of some of the techniques of model-based recognition and background material on the method we use: geometric hashing.

[2]Focal length, aspect ratio and principle point.

This section opened with Ullman's structure from motion theorem. In truth, the theorem was originally due to Kruppa, sixty-four years earlier, and rediscovered by Ullman. This is just one of several cases in which results that have only recently been obtained in the vision community, have been known for many years by photogrammetrists. A fascinating potted history of photogrammetry, from a projective geometry viewpoint, is given in [9], whilst [27] explores the relationship between photogrammetry and machine vision.

2.2 The Fundamental Matrix

In 1981 Longuet-Higgins published a seminal paper [39], in which he described a method for the recovery of the structure of a scene from eight point correspondences. By establishing constraints on the relationship between the two sets of image coordinates, their corresponding 3D points, and the optical centres of the cameras, a 3×3 matrix is defined, the *essential matrix* $\mathbf{E}$, which conveniently encapsulates the epipolar geometry (see section 3.2) of the imaging arrangement. Using homogeneous coordinates (see Appendix A) the essential matrix and a pair of corresponding points p and p' in the two images are related as follows:

$$\mathbf{p}^{'\mathbf{T}}\mathbf{E}\mathbf{p} = \mathbf{0} \tag{2.1}$$

This equation, known as the linear criterion or the Longuet-Higgins relation, is linear and homogeneous in the elements of $\mathbf{E}$. Thus, given a set of at least eight point correspondences (hence the name), the essential matrix can be solved for, up to a scale factor. In the original paper, it is shown how $\mathbf{E}$ can be factorised to obtain a pair of camera matrices, and hence the structure of the scene recovered. There is the implicit assumption in Longuet-Higgins' description that the cameras are calibrated - at least the focal length and principal point are known.

2.2.1 Fundamental v. Essential Matrix

There appears to have been some confusion amongst researchers about the distinction between the essential and *fundamental* matrices. The 8-point algorithm can be used to compute either of them, and equation 2.1 applies equally to essential and fundamental matrices i.e. if $\mathbf{F}$ is a 3×3 fundamental matrix, then

$$\mathbf{p}'^{\mathbf{T}}\mathbf{F}\mathbf{p} = \mathbf{0} \tag{2.2}$$

The question is therefore: when does the 8-point algorithm produce an essential matrix and when does it produce a fundamental matrix? The answer to this question lies in the camera calibration. If the images are formed by projection onto the unit sphere, then the matrix is the product of an orthogonal matrix and an anti-symmetric matrix and it is therefore an essential matrix [13]. In the case of general projection, the matrix $\mathbf{A}$ of intrinsic camera parameters transforms the image into the image that which would have been produced by projection onto the unit sphere. This gives the relation:

$$\mathbf{F} = \mathbf{A}^{-1T}\mathbf{E}\mathbf{A}^{-1} \tag{2.3}$$

Thus, when we are dealing with uncalibrated images, the 8-point algorithm is used to recover the fundamental matrix. The essential matrix can be obtained if the images are taken with a calibrated camera. The two matrices can also be characterised as follows: the essential matrix has zero determinant and its two non-zero singular values are equal, thus it depends on five independent parameters. The fundamental matrix is singular and has rank two. As such, it depends on seven independent parameters. Other properties of the essential matrix have been investigated in [31] and [42].

2.2.2 Calculating the Fundamental Matrix

The main attraction of the 8-point algorithm is its conceptual and numerical simplicity, the fundamental matrix being obtained from a simple set of linear equations. Unfortunately, the 8-point algorithm is susceptible to noise in the matched feature data; poor localisation and false matches causing the algorithm to fail. Until recently, it was generally held that this problem rendered the 8-point algorithm useless for any practical application, and research has concentrated on developing other alternatives of computing the fundamental matrix, all of them more complex than the 8-point method.

However, in [26] a *normalising* technique was presented, that claimed to improve the performance of the 8-point algorithm to a level as good as, and even surpassing, that of these alternatives. The rationale was that the poor performance of the eight-point algorithm is wholly due to implementations which do not take account of the numerical considerations involved, particularly the conditioning of the set of linear equations being solved. The proposed solution to this problem is a simple normalisation (scale and translation) of the matched point coordinates prior to processing. Numerical analysis demonstrates that this transformation leads to a significant improvement in the conditioning of the system [26]. A detailed description of the normalisation method is given in Chapter 3. Here we give a brief overview of some of the alternative methods that have been proposed.

Implementations of the 8-point algorithm attempt to minimise $\min_{\mathbf{F}} \sum_i (\mathbf{p_i'^T} \mathbf{F} \mathbf{p_i})$, which is based on the linear criterion. Typically this is solved either using a closed form solution, setting one of the coefficients of $\mathbf{F}$ to 1, or via a least-eigenvector method, both described in section 3.5. In [40] these linear methods are compared against a number of non-linear criteria, for example minimising the distance of a point from its corresponding epipolar line.

In [62] it is shown that the fundamental matrix can be estimated from the image correspondences of only seven 3D points in general position. First, the correspondences are used to obtain a non-unique solution to the standard system of equations formed from the linear criterion. A further solution of a cubic equation is then necessary to determine the fundamental matrix that corresponds to the given point configuration. The idea here is to use the least possible number of points in determining the fundamental matrix, to reduce the likelihood of including outliers.

An in-depth discussion of the application of robust parameter estimators to calculate the fundamental matrix, is given in [60]. Methods considered here include, RANSAC (see appendix B), the Hough transform [33] and M-estimators. Standard least squares attempts to minimise the sum of the squares of the residuals $\sum \mathbf{r^2}$ i.e. the difference between the observed and fitted data. M-estimators replace $\mathbf{r^2}$ by other functions of the residuals $min \sum \rho(\mathbf{r_i})$, where ρ is a symmetric positive-definite function.

In [8] Boufama describes a completely new approach for computing the fundamental matrix based on *virtual parallax*. This method still requires at least eight point matches, but instead of computing the fundamental matrix directly, it relies on estimating the position of an epipole and a 2D homography. One advantage is that this method implicitly constructs a fundamental matrix that is of rank 2.

Unfortunately, errors in feature localisation and matching are not the only possible causes of problems in fundamental matrix calculations. In the original paper on the 8-point algorithm [39], it is remarked that certain configurations of the eight points will cause the algorithm to fail, due to linear dependencies entering the computation. Examples of such configurations are; four of the points being in a straight line, seven points in a plane, or eight points at the vertices of a cube. A more complete analysis of the problem of degeneracy is given in [62]. Torr defines degenerate configurations of 3D points as those whose resulting image correspondences fail to define a unique epipolar transform (section 3.2). Thus there exist two or more linearly independent fundamental matrices which encapsulate the epipolar geometry, and scene structure cannot be recovered unambiguously. An algorithm for detecting such degenerate configurations, which is robust to the presence of outliers, is described.

A complete review of the issues involved in fundamental matrix theory was recently published in [41].

2.3 Uncalibrated Reconstruction

In 1992 Faugeras published a ground-breaking paper [12] that proposed a technique for recovering *projective* structure of a scene, from a set of matched points in a pair of uncalibrated images. The idea is to constrain the form of the two camera matrices. This is achieved by fixing the coordinates of five of the 3D points to be the standard projective basis, and making appropriate coordinate assignments for their corresponding image points. As a result the camera matrices can be represented as functions of just two arbitrary parameters.

The exact determination of these parameters requires the coordinates of the epipoles and hence, computation of the fundamental matrix. With this done, the camera matrices are fully specified and the locations of the 3D points can be recovered, relative to the coordinate system defined by the five points. Thus the reconstruction is correct up to a projective transform of the real structure. The problem with this method is in the reliance on the accuracy of the chosen basis features. Mismatched or poorly localised basis features will impair the quality of the projective reconstruction.

Just one month later, Hartley published his paper [23], which achieved the same results, in a different manner and without the reliance on basis points. In this case, the projective reconstruction is achieved through the analysis of the fundamental matrix. Just as the essential matrix can be factorised to obtain camera matrices, so too can the fundamental matrix. In the latter case however, it is shown that the factorisation is not unique, and that the camera matrices can be transformed by an arbitrary projectivity and still be a valid factorisation. Hence, once again, the recovered structure and camera locations differ from the real ones by a projective transform. The projective reconstruction method employed in Chapter 4, is based around this technique.

The standard basis method above is one of several projective reconstruction techniques compared in a recent paper by Rothwell [52]. Rothwell places the basis method in a group of what he calls *explicit* reconstruction algorithms. The goal for each of these is to compute a pair of camera matrices from a set of image correspondences. Two other explicit methods are described. One estimates structure by computing the intersection of camera rays on which the 3D points must lie. Another uses singular value decomposition to obtain estimates for the camera matrices which are consistent with the epipolar geometry. Two *implicit* techniques are described, which compute structure using three-dimensional invariants of the camera configuration and point sets. All the methods described in this paper assume knowledge of the *weak calibration* between the two cameras; i.e. the fundamental matrix. Rothwell concludes that the most reliable reconstructions are obtained using camera matrices derived via singular value decomposition, which is the method we use in Chapter 4.

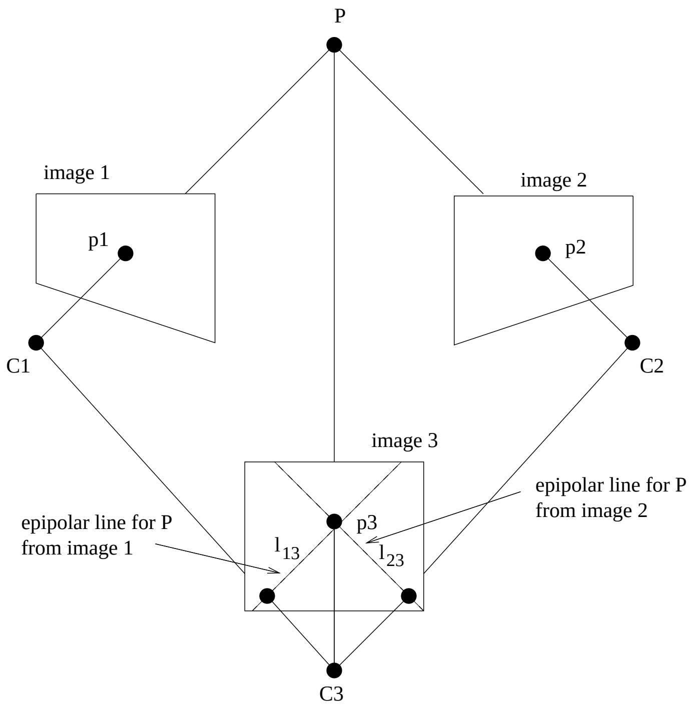


Figure 2.1: Transfer.

2.3.1 Other Applications of the Fundamental Matrix

Transfer

One simple application which follows naturally from the definition of the fundamental matrix is in the area of automatic feature matching. The fundamental matrix maps a point in one image to its corresponding epipolar line in the other. This constraint can be used to reduce the search space for the matching point. This idea extends to three images. The observation of the point P in image 1 can be used to generate an epipolar line l_{13} for P in image 3; similarly, observation of P in image 2 can be used to generate a second epipolar line l_{23} for P in image 3. As shown in figure 2.1 the intersection of the epipolar lines l_{13} and l_{23} uniquely defines where P must appear in image 3. This concept is known as *transfer*.

Self-Calibration

A new approach to calibrating cameras, called *self-calibration* was presented by Faugeras et al in [13]. In contrast to existing techniques, which rely on calibration objects [64], all that is required is a set of point matches tracked over three or more camera displacements. The displacements are used to calculate the fundamental matrix, using the non-linear minimisation of the image plane distance of a point from its corresponding epipolar line, mentioned earlier. The epipolar transform (section 3.2) is derived from the fundamental matrix and used to solve for the coefficients of the absolute conic, from which the camera intrinsic parameters are obtained[9].

The original authors [13] note that the precision of feature localisation required to obtain reasonable calibration results is at the limit of even the best feature detectors. Nevertheless, the method has been used in attempts to calibrate cameras automatically, and thus to recover Euclidean structure [25, 2].

2.4 Sequence-Based SFM

In this section we present a brief review of some current related research into reconstruction from image sequences.

The VSDF

The *Variable State-Dimension Filter* (VSDF) is not really a reconstruction system. Rather, it is a recursive estimation algorithm which has been applied to the problem of structure from motion [43]. In fact, one of the benefits of the VSDF is that it can be applied to any problem that can be formulated as a suitable measurement equation, for example [44].

The algorithm was designed specifically for real-time applications. Having computed optimal estimates for the structure and motion over a small number of initial images, the recursive part of the algorithm takes over and recomputes sub-optimal estimates, using new image data. An example reconstruction[3] is shown in figure 2.2.

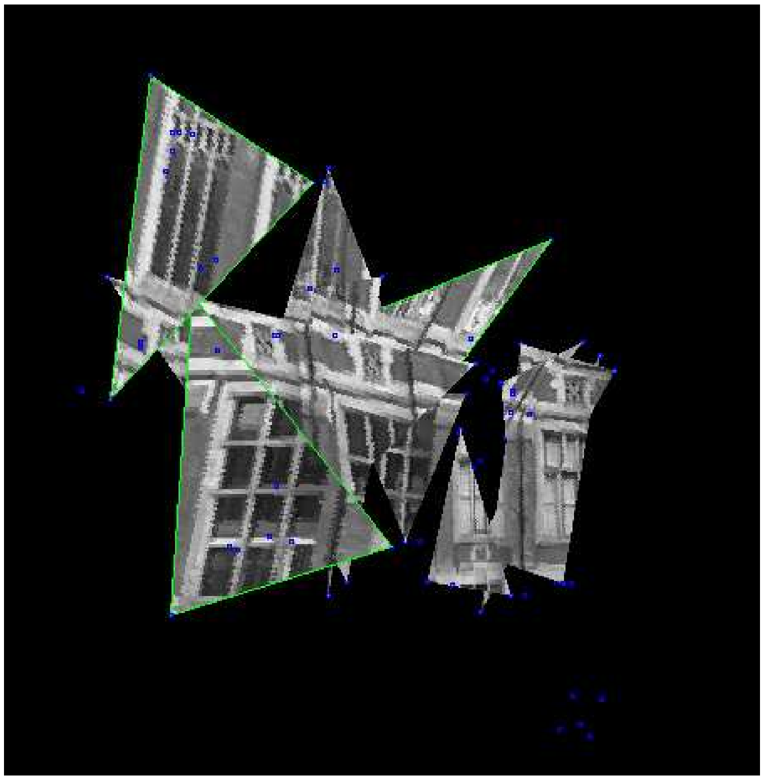


Figure 2.2: Example reconstruction using VSDF.

Vanguard

Vanguard[4] is a project being developed by a European Union consortium, including the Robotics Research Group at Oxford University. The goal is automatic 3D model building from long uncalibrated monocular image sequences, and the use of these models for rendering scenes in telepresence applications. This entails extracting both geometry and surface descriptions (reflectance) at a level suitable for high quality graphical rendering.

The system uses a robust tracking algorithm for corner and line segment features, based on the *trifocal tensor* [61]. The trifocal tensor performs a similar role for three views as the fundamental matrix does for two: it encapsulates all geometric constraints between the three views, that are independent of scene structure. Given point correspondences in two images, the trifocal

[3]This reconstruction was obtained using software kindly supplied by the VSDF author, Phil McLauchlan.

tensor determines the position of the point in the third. This is similar to *transfer*, described earlier in this chapter, but more robust.

The trifocal tensor is computed from an initial set of feature matches in three images but for future images does all the matching itself. Camera matrices are generated from the tensor for these images, and used to instantiate 3D point and line structure. As each new image is processed, matches between it and the previous image provide a correspondence between existing 3D structure and new features, enabling a camera matrix for the new image to be obtained, thus determining the new camera position relative to the existing world coordinate system. Existing structure estimates are updated using an Extended Kalman Filter. Figure 2.3 shows some sample data and results obtained [59].

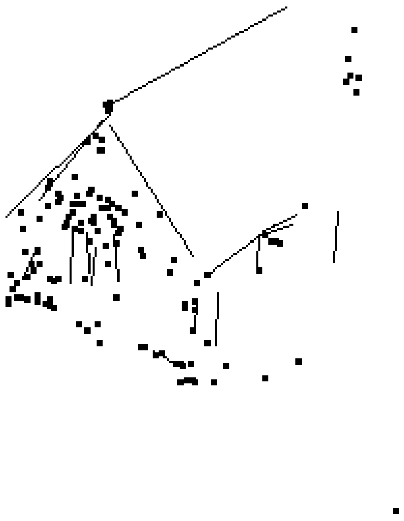
The Factorisation Method

The factorisation method, originally due to Tomasi and Kanade [58], is a batch method for recovering the structure and motion of an object from an image sequence. Point features are tracked through the sequence and used to construct a *measurement matrix*. Structure and motion are obtained by factorising the measurement matrix, using singular value decomposition [50]. It is implicitly assumed that camera intrinsic parameters are known.

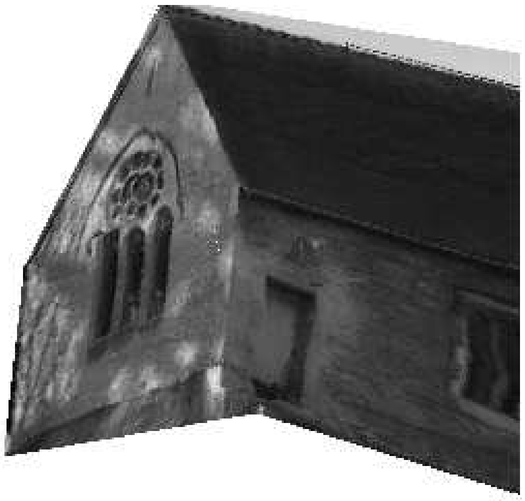
The method has been through many stages of development, starting with simple planar motion, then moving on to arbitrary motion in 3D with 2D images obtained under the orthographic camera model. The orthographic model was too simplistic to be of any practical use, being unable to model even the effect of distance on image size, the scaling effect. Further updates followed through the scaled orthographic and paraperspective models [49] to the projective model [48]. While still in batch form the algorithm was unsuitable for use in real-time applications, and as a result a sequential version was developed [46], but this needed most of the feature points to be visible in each frame, which is not feasible in an extended environment.

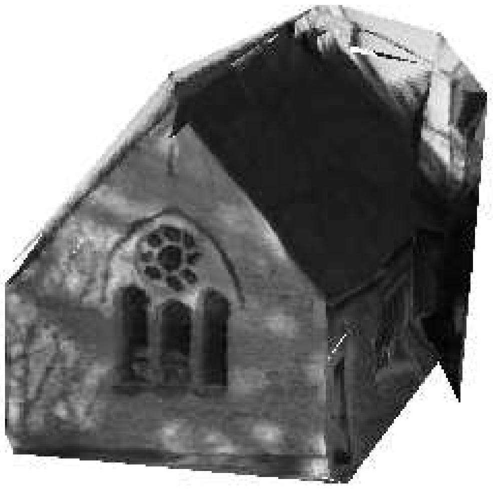
The latest addition to the factorisation approach was in a recent paper by Held [28], which described an incremental *windowed factorisation* method. At each step Held applies a version of Morita's sequential factorisation algorithm, to obtain a new shape matrix. This is then related

Sample Image



Recovered Structure





Two novel rendered views.

Figure 2.3: Example data from the VANGUARD project.

to the shape matrix at the previous step by computing an affine transform between their shared shape points, then applying it to the points which have just been lost. Thus the scene structure is recovered incrementally as the 'factorisation window' moves through the image sequence.

The major differences between our method and Held's are that he has gone back to using the orthographic projection model, whereas we use full perspective, and he makes no attempt to register between structure recovered at different times.

2.5 Model Based Recognition

The structure matching required within our framework is most closely analogous to model-based approaches to object recognition. As background to our choice of matching strategy, this final part of the chapter reviews the major approaches in this area.

Object recognition is one of the most fundamental of all problems studied in machine vision. The aims of an object recognition system are twofold: to detect the presence of a given object, and to establish its location and orientation. The latter of these is known as *pose determination*. The pose of an object can be expressed in terms of the rigid transformation required to rotate and translate to the object position from the origin of the given coordinate system.

The most predominantly used approach to object recognition is *model-based*, which relies upon the construction of explicit representations (models) of the geometric shape of the objects which are to be recognised. In this section we provide a brief overview of a number of different model-based recognition schemes, finishing with the geometric hashing paradigm, which is the method used in this work. For more detail the reader is referred to the general surveys on model-based recognition in [5] and [10].

In general, model-based recognition is a two-step process: *hypothesis generation* and *verification*, for example [51]. In the first stage, a number of likely candidate objects and poses are singled out for further investigation. These are examined more closely in the second stage and incorrect candidates are eliminated. Hypothesis generation involves matching subsets of model and scene features, for example corner points and edges. Verification requires the computation of the transformation between the model and scene, in an attempt to induce more feature correspondences and thus accept or reject the candidate match. In model-based recognition, pose determination involves computing the transformation between the model and the scene.

Note that we use the terms *model* and *scene* without referring to their dimensionality, or the nature of the transformation between them. For example, the model and scene could both be image features, related by a 2D transform [37]. Equally, they could both be sets of structure points, related by a 3D transform, for example this work and [14]. The model and scene could even be of different dimensionality, as in the case of recognising 3D objects from 2D images [16, 6].

Alignment

In an *alignment* scheme [32], candidate poses are hypothesised, based on the correspondence of a minimal number of model and scene features. Each minimal set of correspondences is just enough to determine a unique pose, which is then verified.

Pose Clustering (Hough Transform)

Pose clustering is similar to alignment but uses a more intelligent, rather than exhaustive, approach to selecting candidate poses for verification. The idea is each pairing of model/scene features tallies a vote for the pose they determine. Correct poses should be voted for many times by different pairings of features and only high-scoring poses are considered for verification. In practice, things are not quite as simple as that, since correct feature pairings will generally determine poses that are similar, but not identical, due, for example, to measurement error.

The solution is to look for clusters of poses (hence the name). The usual way of doing this is by creating a parameterised *pose space* in which to tally the votes. For example, a 2D affine transform can be represented by by six independent parameters. Thus voting is performed in a six-dimensional pose space, represented by a multi-dimensional array, with all elements, or *bins*, initially zero. Although each parameter may take on continuous values, each dimension of the pose space is quantised, resulting in a set of six-dimensional volumes of admissible poses. The parameters of each candidate pose are calculated and used to increment the vote count at the corresponding bin. When all votes have been cast, the pose clusters correspond to those bins containing more than some given number of votes. A pose at the centre of each cluster is passed on to the verification stage.

Pose clustering methods are generalisations of the Hough transform [3], which was originally used for the detection of straight lines in images [30]. A survey of the Hough transform is given in [33].

Interpretation Tree Search

Alignment and pose clustering techniques produce candidate poses from correspondences between model and scene features, eliminating incorrect candidates by direct verification. In contrast to this, *interpretation tree search* methods [21, 19] attempt to assign model features to all scene features, in all feasible combinations. Such assignments are called interpretations and the method involves performing a tree search to generate all of the combinations of interpretations as required.

Nodes in the first level of the tree contain assignments for the first scene feature. Those at the second level contain explicit assignments for the second scene feature, and an implicit assignment for the first, and so on for each level of the tree. Likewise, each branch corresponds to a different model feature. Each node in the tree represents a partial match between model and scene features and the path to the node from the root of the tree gives all the correspondences in the partial match. Finally, each leaf node defines a complete candidate match (interpretation) between the model and the scene.

As described, the method is infeasible, since there is a vast number of interpretations for even a moderately complex model. The idea is to eliminate the need to verify many false matches by *pruning* the tree. This is possible because false matches can be identified without needing to assign all the pairings. Each time a new node is to be added to the tree a set of fast consistency checks are performed against all nodes on the path back to the tree root. For example, if attempting to match a set of planar surface patches against a model in 3D, the angle between the surface patch normals could be examined to ensure it falls between some range of values, this range having been obtained by preprocessing the model before the search begins. Thus, the tree is pruned below any partial interpretation that is found to be inconsistent.

Indexing and Geometric Hashing

An alternative model-based recognition paradigm has been proposed, known as *indexing*. All indexing schemes share a common approach to the recognition problem. They compute invariants [47] from scene features and use them to index a look-up table containing references to model objects. The look-up table returns a weighted set of candidate matches, for verification. Whereas the methods described previously must be applied separately for each of the models they should recognise, indexing schemes attempt to recognise all models *simultaneously*. This has obvious efficiency benefits when the size of the model database is large.

Indexing techniques have been applied to a number of problems. For example, in [45] Mohan describes a method for recognising 3D objects from a 2D image sequence, assuming the weak perspective camera model. Object models are acquired automatically from the image sequence by tracking features through at least three frames, and extracting Euclidean (similarity) invariants [68]. In [54, 56, 55] Rothwell et al. describe the development of a complete model-based recognition system (LEWIS). Their system also acquires its models directly from images, this time using projective invariants to index the look-up table.

Perhaps the best-known indexing method is *geometric hashing*, originally developed for the task of recognising flat rigid objects [35, 34, 36] from images, using an affine approximation to the full perspective camera. However, the same approach can be used for many recognition problems involving a variety of transformations in two and three dimensions [37]. A full description of the method is given in Chapter 5, where it has been applied to the problem of recognising familiar segments of 3D structure, as they are recovered by the reconstruction system. Geometric hashing has an additional property which makes it ideal for our purposes; the nature of the information stored in the hash table means that the pose of an object is known as soon as it is recognised.

There has been some discussion about the susceptibility of geometric hashing to sensor error [20]. A simple technique for taking errors into account is shown in [70], which involves tallying votes in a region of the hash table, rather than at a single, indexed location. Other additions to the original geometric hashing method have been developed. For example, in [63] Tsai presents a system which uses invariants computed from line features, under the assumption that these can be acquired from images with greater accuracy then points.

In [14] an interesting application of geometric hashing is described, that of matching protein molecules. The system uses a two-atom basis and indexes the hash table using the lengths of the sides of the triangle the basis forms with each of the remaining atoms. This approach reduces the complexity of the process from $O(n^4)$ to $O(n^3)$ at the expense of a non-unique representation of the atoms[4]. Constraints on the allowable lengths of the triangle sides mean that not all basis/atom pairs need to be considered.

Finally, a new technique has recently been proposed, called *enhanced geometric hashing* [38]. This extends the basic method in two ways. First of all, the use of *quasi-invariants* [7] extracted from connected segments, reduces the number of invariants that have to be computed. Secondly, the voting scheme is augmented through the use of pose clustering. Each candidate pose, obtained via the normal voting procedure, defines a geometric transform, which is parameterised and used to vote in the pose space, as described above. Clusters form around coherent poses and the matched model is taken to be the one whose pose space contains the highest density cluster.

[4]Where n is the number of points used to define a basis for the transformation being considered.

Chapter 3

Acquiring the Fundamental Matrix

3.1 Introduction

Since the pioneering work of Faugeras [12] and Hartley [24], there has been a great deal of research into developing methods for recovering the structure of a scene from images taken with uncalibrated cameras. A recurring theme in this work, and in the method we describe in the next chapter, is the requirement for the accurate estimation of the *fundamental matrix*. The fundamental matrix is of vital importance in the the analysis of pairs of uncalibrated images, because it encapsulates all the information about camera motion, camera parameters and epipolar geometry that can be obtained from a set of point correspondences. The most simple method of computing the fundamental matrix is the *8-point algorithm*, due to Longuet-Higgins [39]. This involves the solution of a set of linear equations derived from the *linear criterion*, which relates the fundamental matrix, image points and corresponding epipolar lines.

A common criticism of the 8-point algorithm is that it is very sensitive to noise in the measured image features. In fact, the prevailing view is that this flaw renders the 8-point algorithm useless for any practical application [40]. This has led to the development of a number of alternative methods of solution, as discussed in Chapter 2. These are, without exception, more complicated than the 8-point algorithm, involving iterative schemes [25] or non-linear minimisation [13, 40]. However, in a recent paper [26], Hartley described a novel *normalisation* technique, which claimed to improve the performance of the 8-point algorithm to a level as good as (and even surpassing) that of the more complex methods.

This technique was based on numerical analysis of a particular method of implementing the 8-point algorithm. One of the aims of this chapter is to determine what effect, if any, the normalisation process has when a different implementation method is used. Hartley described a series of experiments, in which the fundamental matrix was computed using different sized subsets of the matched image points. For each subset size, several trials were performed, each time randomly selecting the points used in the calculation. Given a measure of the accuracy of a fundamental matrix, results were presented in terms of subset size versus the *median* accuracy value obtained over all the trials. We have carried out similar experiments, but in contrast, our results show the effects of normalisation and subset size on the *best* accuracy values. There is the additional problem of determining the best subset of matched points to use to estimate the fundamental matrix. Rather than simply choosing the one that gives highest accuracy value over all the trials, we employ a more efficient, elegant solution based on the RANSAC parameter estimation paradigm [15].

This chapter describes our implementation and experiences in using the normalised 8-point algorithm to compute the fundamental matrix. We begin with an overview of epipolar geometry and derive the linear criterion on which the 8-point algorithm is based. This is followed by a brief description of some of the important properties of the fundamental matrix. Next we describe two different methods of solving the linear criterion equations. We then outline the normalisation process itself and explain how it can help improve the accuracy of the solution of these equations. Finally, we present the results of a series of experiments to compute the fundamental matrix, using a variety of real and synthetic data, and show how we can embed the normalised 8-point algorithm into the RANSAC paradigm.

Appendices A and B provide essential background material on projective geometry and the RANSAC paradigm, respectively.

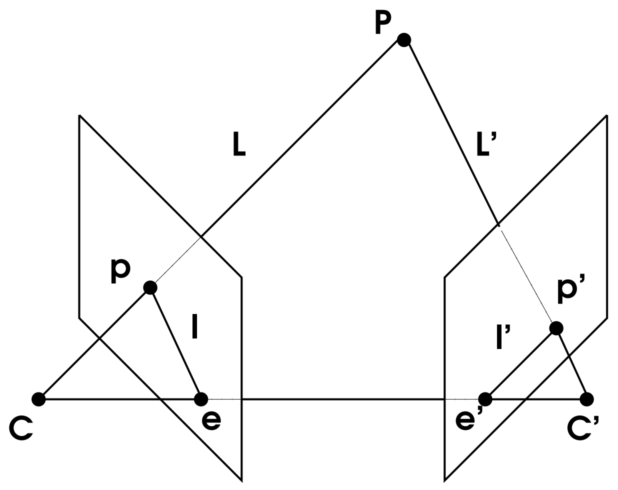


Figure 3.1: Epipolar Geometry: Two images of a point P are taken by a moving camera, whose optical centre is at positions C and C'. The plane through CPC' is the epipolar plane for P. This plane intersects the images along the two epipolar lines l and l'.

3.2 An Overview of Epipolar Geometry

Figure 3.1 illustrates the epipolar geometry. Suppose we have a pair of images of a scene, taken with either a single moving camera or a stereo pair. The 3D locations of the camera optical centres are at C and C'. The 3D scene point P is projected along rays L and L' onto the image planes at p and p', respectively. Projection of C along the line joining the two optical centres, gives a point e' in the second image, known as the *epipole*. Similarly, projecting C' gives the epipole e in the first image. Note that, for reasons of clarity, the imaging arrangement we have shown results in the epipoles lying within the bounds of the two images. In practice this is not necessarily the case, indeed the line between the two optical centres may only intersect the image planes at infinity.

The 3D point P and the two optical centres define the *epipolar plane* for P. This intersects the image planes along the *epipolar lines* l and l'. It can be seen that l is the projection of the ray L' onto the first image and l' is the projection of the ray L onto the second image. This property has important implications when attempting to match points in the two images. If the 3D point P has been observed at position p in the first image, then P must lie somewhere on the ray L. This, in turn, constrains the position of p': it must lie somewhere along the projection of L in the second image i.e. on the epipolar line l'. Consequently, given a mechanism for calculating the epipolar line corresponding to a point in one image, the search space for the matching point can be dramatically reduced from the whole image to just a line. The fundamental matrix provides such a mechanism.

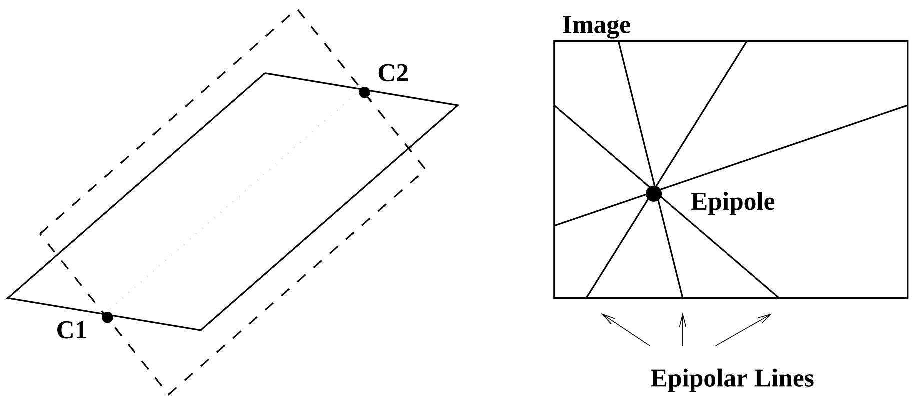


Figure 3.2: Each 3D point defines a plane with the camera optical centres. These planes intersect the images, forming two pencils of epipolar lines, with intersections at the epipoles

Consider figure 3.2 and imagine a set of 3D points at general positions in the scene. Each point forms a different epipolar plane with the optical centres. As we have seen, each plane forms an epipolar line at its intersection with the image planes, thus multiple planes lead to the formation of a pencil of lines, which intersect at the epipole in either image.

Let Π be any such plane, then Π projects to an epipolar line l in the first image and l' in the second. The correspondences $\Pi \bar{\wedge} l$ and $\Pi \bar{\wedge} l'$ are homographies between the two pencils of epipolar lines and the pencil of planes through the optical centres. [1] It follows that the correspondence $l \bar{\wedge} l'$ is also a homography. This homography is the epipolar transform. It is determined by the coordinates of the two epipoles and three $l \bar{\wedge} l'$ correspondences. It follows that the epipolar transform depends on seven independent parameters. In fact, it can be shown ([13, 41]) that the epipolar transform determines, and is itself determined by, the fundamental matrix.

[1] We use the symbol $\bar{\wedge}$ to denote homographic correspondence.

3.3 Derivation of the Linear Criterion

In this section we describe an alternative derivation of the linear criterion, due to Hartley [24]. This projective geometry version fits better with the uncalibrated/projective nature of the fundamental matrix than the calibrated/Cartesian method used by Longuet-Higgins in the original paper.

Given the same imaging arrangement as in figure 3.1, we fix the first camera position, C, at the origin of our object space coordinate system. The second camera, C', is located at some displacement from this. We can represent the two cameras by the 3×4 transformation matrices they use to project from 3D object space coordinates to the 2D image planes (see Appendix A). We can assign the two camera matrices as follows:

$$\mathbf{C} = (\mathbf{I}|\mathbf{0}) \qquad \text{and} \qquad \mathbf{C}' = (\mathbf{R}| - \mathbf{R}\mathbf{t}) \tag{3.1}$$

where we have partitioned the 3×4 matrices into a 3×3 left sub-matrix and a 3×1 column vector, $\mathbf{I}$ is the identity matrix, $\mathbf{R}$ is a 3D rotation matrix and $\mathbf{t} = (t_x, t_y, t_z)$ is a 3D translation matrix. These two cameras matrices project a 3D image point $\mathbf{P} = (X, Y, Z, 1)^T$ as follows:

$$\mathbf{p} = \mathbf{C}\mathbf{P} \Leftrightarrow (u, v, w)^T = (\mathbf{I}|\mathbf{O})(X, Y, Z, 1)^T \tag{3.2}$$

and

$$\mathbf{p}' = \mathbf{C}'\mathbf{P} \Leftrightarrow (u', v', w')^T = (\mathbf{R}| - \mathbf{R}\mathbf{t})(X, Y, Z, 1)^T \tag{3.3}$$

Now, as we saw in the previous section, given a point p in the first image, the corresponding point, p' in the second image is constrained to lie on the epipolar line, l', which is the projection of L by C'. Another way of looking at L is as the set of all 3D points which project onto p under C. We now choose two points from this set; the camera origin $(0, 0, 0, 1)^T$ and the point at infinity $(u, v, w, 0)^T$. The coordinates of the projections of these points in the second image are $-\mathbf{R}\mathbf{t}$ and $\mathbf{R}(u, v, w)^T$ respectively. The homogeneous equation of the epipolar line, $\mathbf{l}'$, passing through these two points can now be recovered using the cross product:

$$(a, b, c)^T = \mathbf{R}\mathbf{t} \times \mathbf{R}(u, v, w)^T = \mathbf{R}(\mathbf{t} \times (u, v, w)^T) \tag{3.4}$$

where $\mathbf{l}' = (a, b, c)$ represents the line $au + bv + cw = 0$. If we now define the anti-symmetric matrix $\mathbf{S} = [\mathbf{t}]_\times$ as:

$$\begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{3.5}$$

the properties of the cross product allow us to rewrite equation 3.4 as:

$$(a, b, c)^T = \mathbf{RS}(u, v, w)^T \tag{3.6}$$

We set $\mathbf{F} = \mathbf{RS}$, giving $\mathbf{F}$ as the 3×3 fundamental matrix and can now write:

$$\mathbf{l}' = \mathbf{Fp} \tag{3.7}$$

Thus, given a point in one image, the fundamental matrix allows us to compute the corresponding epipolar line in the other. It is also clear that, since, by definition, the corresponding image point, p' belongs to epipolar line l', we obtain the linear criterion:

$$\mathbf{0} = \mathbf{p}'^{\mathbf{T}}\mathbf{Fp} \tag{3.8}$$

It is worth noting that by reversing the roles of the two images in this derivation, the fundamental matrix is switched to its transpose. That is, the following relations also hold:

$$\mathbf{l} = \mathbf{F}^{\mathbf{T}}\mathbf{p}' \qquad \text{and} \qquad \mathbf{0} = \mathbf{p}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}\mathbf{p}' \tag{3.9}$$

3.4 Other Properties of F

It has already been mentioned that $\mathbf{F}$ determines and is determined by the epipolar transform, which has only seven independent parameters. Although the fundamental matrix has nine elements, it too has only seven independent parameters. We can account for one of the degrees of freedom by noting that $\mathbf{F}$ is only defined up to a scale factor. That is, applying an arbitrary scale to $\mathbf{F}$ will have no effect on the results of equations 3.7and 3.8.

The other degree of freedom is taken care of by the fact that the fundamental matrix must have rank 2. To see why this is so it may be necessary to consider figure 3.1 once again. As we know, application of the fundamental matrix to a point in one image, yields the corresponding epipolar line in the other, but what if the point in question is one of the epipoles? For example, $\mathbf{Fe} = \mathbf{l_e}$. Geometrically, l_e is the projection of the ray $\langle C, e \rangle$ onto the second image, but by construction this line is reduced to a point, the corresponding epipole e'. Thus, we have the following property:

$$\mathbf{Fe} = \mathbf{F^T e'} = \mathbf{0} \tag{3.10}$$

Consequently, $\mathbf{F}$ is singular, has zero determinant and its rank *must be* less than or equal to 2. In general it is of rank 2. It is not possible for the rank to be 1, since this implies that the line between the optical centres belongs to the intersection of the image planes [41].

3.5 Implementing the 8-Point Algorithm

Equation 3.8 is linear and homogeneous in the nine unknown coefficients of the fundamental matrix. Thus, in general, given a set of eight point matches $p_i \leftrightarrow p_{i'}$ in the two images, we will be able to obtain a unique solution for $\mathbf{F}$, up to a scale factor. Taking one such pair of points $\mathbf{p} = (u, v, 1)$, $\mathbf{p'} = (u', v', 1)$ and labelling the coefficients of $\mathbf{F}$, gives an expanded linear criterion equation:

$$(u', v', 1)^T \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} (u, v, 1) = 0 \tag{3.11}$$

Multiplying out and rearranging in terms of the known coordinates of $\mathbf{p}$ and $\mathbf{p'}$ gives:

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0 \tag{3.12}$$

Each pair of matched points gives one such equation. With more matches, we can build up a set of homogeneous linear equations $\mathbf{A}\mathbf{f} = \mathbf{0}$:

$$\begin{pmatrix} uu' & vu' & u' & uv' & vv' & v' & u & v & 1 \\ \multicolumn{9}{c}{\dotfill} \\ \multicolumn{9}{c}{\dotfill} \\ \multicolumn{9}{c}{\dotfill} \\ \multicolumn{9}{c}{\dotfill} \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ \cdots \\ \cdots \\ \cdots \\ \cdots \end{pmatrix} \tag{3.13}$$

The process of generating and solving the above system of equations *is* the 8-point algorithm. In practice, we are given many more than just eight matches, resulting in an over-determined set of equations.

In order for there to exist a non-trivial solution to $\mathbf{A}\mathbf{f} = \mathbf{0}$, the matrix $\mathbf{A}$ must be rank-deficient, that is, although $\mathbf{A}$ has nine columns, its rank will be at most eight. This is generally true if we are dealing with perfect data, but in practice inaccuracies in the matched points lead to $\mathbf{A}$ having full rank. In this case we seek a least-squares solution to the linear criterion equations. We will now look at two different methods of obtaining this solution.

3.5.1 Solution Via Singular Value Decomposition - SVD

Since $\mathbf{F}$ is only defined up to a scale factor, we can fix one of its coefficients to a known value and then compute a closed-form solution to the linear criterion equations. Numerical analysis of *non-linear* solutions has shown that the choice of this normalising coefficient is not arbitrary: it should be one of the first six [40]. By setting $f_{13} = 1$ we obtain the following modified system of linear equations $\hat{\mathbf{A}}\hat{\mathbf{f}} = -\mathbf{u}'$:

$$\begin{pmatrix} uu' & vu' & uv' & vv' & v' & u & v & 1 \\ \multicolumn{8}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \multicolumn{8}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \multicolumn{8}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \multicolumn{8}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \begin{pmatrix} -u' \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix} \tag{3.14}$$

This system can then be solved, using Singular Value Decomposition [50], which is known to give the best estimate for the remaining coefficients of $\mathbf{F}$, in a least-squares sense. This is perhaps the simplest method for computing the fundamental matrix, and possibly the one which is most prone to the effects of noise and outliers in the matched features. Although the normalisation transform, discussed in the next section, was not designed specifically with this approach in mind, it will be interesting to see what effects it has. In the sequel we will refer to this method of solution as **SVD**.

3.5.2 Solution Via Eigenvector - EIG

Another, more popular, method of computing the fundamental matrix is to formulate the linear criterion equations as a classic minimisation problem and seek the vector $\mathbf{f}$ which minimises:

$$\min_{\mathbf{f}} ||\mathbf{Af}|| \qquad \text{subject to} \qquad ||\mathbf{f}|| = \mathbf{1} \tag{3.15}$$

where $||.||$ indicates the Frobenius norm [2]. It is well known that the solution to this problem is the unit norm eigenvector of $\mathbf{A^T A}$, corresponding to the smallest eigenvalue. In our implementation we solve this problem by first reducing $\mathbf{A^T A}$ to tri-diagonal form, via the Householder method. The eigenvalues and eigenvectors can then be recovered using the QL algorithm. See [50] for the details. From now on we will refer to this solution method as **EIG**.

[2] The square root of the sum of squares of the coefficients. For a vector, the Frobenius norm is equivalent to the two norm.

3.5.3 Enforcing the Rank Constraint

As noted in section 3.4, important properties of the fundamental matrix are that it is singular, has rank 2 and $\mathbf{det}(\mathbf{F}) = \mathbf{0}$. In fact, most applications of $\mathbf{F}$, depend on this *singularity* or *rank constraint*. However, in general, the fundamental matrix obtained using the methods based on the linear criterion will not have these properties, or, to put it another way, the linear criterion cannot express the rank constraint. To see the effect this has, consider a fundamental matrix, $\mathbf{F}$, computed via a linear criterion method, a point $\mathbf{p} = (u, v, 1)$ in the first image and its corresponding epipolar line l'. Furthermore, we denote the coordinate vector of the epipole in the first image by $\mathbf{e} = (e_u, e_v, 1)$ and fix the horizontal and vertical offsets of p from e as x and y, respectively. We can now express $\mathbf{p}$ as:

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} e_u - x \\ e_v - y \\ 1 \end{pmatrix} \tag{3.16}$$

Now, applying the fundamental matrix, gives:

$$\mathbf{l}' = \mathbf{F}\mathbf{p} = \mathbf{F} \begin{pmatrix} e_u - x \\ e_v - y \\ 1 \end{pmatrix} = \mathbf{F}\mathbf{e} - \underbrace{\mathbf{F} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}}_{l_{offset}} \tag{3.17}$$

If $\mathbf{F}$ is singular, $\mathbf{F}\mathbf{e} = \mathbf{0}$, exactly, and the right hand side of the equation simplifies to $\mathbf{l_{offset}}$, which is an epipolar line, as required. However, when $\mathbf{F}$ is non-singular, $\mathbf{l}'$ is the sum of a constant residual vector $\mathbf{r} = \mathbf{F}\mathbf{e}$ and the vector $\mathbf{l_{offset}}$, whose norm is bounded by $\sqrt{x^2 + y^2}||\mathbf{F}||$. It can be seen that as $p \to e$ so $(x, y) \to (0, 0)$ and thus $l' \to r$, which is inconsistent with the principles of epipolar geometry. In fact, the closer p gets to the epipole, the greater will be the error in its corresponding epipolar line. Thus, if the epipole lies within the bounds of the image, then the epipolar geometry encompassed in a fundamental matrix obtained using the linear criterion will be inaccurate.

Fortunately, this problem is not insurmountable. The solution is to correct the estimated fundamental matrix to ensure it has the required rank. To do this we employ a technique developed by Tsai and Huang [65], whereby $\mathbf{F}$ is replaced by the matrix $\hat{\mathbf{F}}$ that minimises:

$$||\mathbf{F} - \hat{\mathbf{F}}|| \qquad \text{subject to} \qquad \mathbf{det}(\hat{\mathbf{F}}) = \mathbf{0} \tag{3.18}$$

This is achieved by first computing the singular value decomposition, $\mathbf{F} = \mathbf{UDV}^T$, where $\mathbf{D}$ is a diagonal matrix $\mathbf{D} = diag(r, s, t)$ such that $r \geq s \geq t$. Setting $\hat{\mathbf{F}} = \mathbf{U}diag(r, s, 0)\mathbf{V}^T$ has been shown to give $\hat{\mathbf{F}}$ as the closest singular matrix to $\mathbf{F}$ under Frobenius norm.

3.5.4 Normalisation

Here we give a brief description of a recently developed technique [26] for improving the accuracy of the fundamental matrix computed using the 8-point algorithm. This approach is based on numerical analysis of the **EIG** method of solving the linear criterion equations. It is theorised that the poor performance of the 8-point algorithm can be attributed to methods of implementation that do not take sufficient account of the conditioning of the set of equations being solved. The condition number, κ of a matrix M is given by

$$\kappa = ||\mathrm{M}||.||\mathrm{M}^{-1}|| \tag{3.19}$$

It plays an important role in the analysis of linear problems. When κ is large, a small change in the data can lead to large variations in the computed solution. Thus, we aim to make κ as small as possible, improving the conditioning of the system of equations and leading to a more stable and accurate solution.

The **EIG** method of solving $\mathbf{Af} = \mathbf{0}$ requires the computation of the unit norm eigenvector of $\mathbf{A^TA}$. As such, the result of Hartley's analysis was the development of a normalising transform to reduce the condition number of the $\mathbf{A^TA}$ matrix, allowing us to obtain a better estimate for $\mathbf{F}$. However, as we will show, the beneficial effect of normalisation is not just limited to the **EIG** implementation, it also improves the results of the **SVD** method.

The normalisation process is straightforward. Prior to generating the linear constraint equations, the coordinates of the matched image points undergo a combined translation and scale transformation such that the centroid of the points is at the origin and the average distance of a point from the origin is $\sqrt{2}$.

The effect of the translation is intuitively obvious. Consider a set of points in a 200×200 image, whose u-coordinates are $101, 102, 103$. Translating by 100 results in $1, 2, 3$. Thus in the untranslated coordinates, the important values are not found until the third significant figure, being obscured by the offset of 100. This has a detrimental effect on the conditioning of $\mathbf{A}^\mathbf{T}\mathbf{A}$. However, this problem can be solved by a simple translation which promotes the coordinates' significant figures.

The theory behind the scaling effect is quite complex and we will not discuss it in any great depth here. In essence, a lower bound for the condition number of the $\mathbf{A}^\mathbf{T}\mathbf{A}$ is derived, based on the *interlacing property*[3] for the eigenvalues of a symmetric matrix and their relationship with the values of the diagonal elements of $\mathbf{A}^\mathbf{T}\mathbf{A}$. Clearly, the magnitudes of these diagonal elements are themselves related to the coefficients in equation 3.12, which, in turn, are determined by the matched point coordinates. Thus, it can be shown that scaling so that the average homogeneous point coordinate is unity will improve the conditioning of $\mathbf{A}^\mathbf{T}\mathbf{A}$.

Note that the translation and scale transforms are computed separately for each image. Each pair of transforms is then combined to give a single transformation matrix $\mathbf{T}$ and $\mathbf{T}'$, for the first and second image respectively. Thus two matched image coordinates $\mathbf{p}$ and $\mathbf{p}'$ are replaced by their normalised versions as follows:

$$\hat{\mathbf{p}} = \mathbf{T}\mathbf{p} \qquad \text{and} \qquad \hat{\mathbf{p}'} = \mathbf{T}'\mathbf{p}' \tag{3.20}$$

thus

$$\mathbf{T}^{-\mathbf{1}}\hat{\mathbf{p}} = \mathbf{p} \qquad \text{and} \qquad \hat{\mathbf{p}'}^\mathbf{T}\mathbf{T}'^{-\mathbf{T}} = \mathbf{p}'^\mathbf{T} \tag{3.21}$$

which, by the linear criterion (3.8), gives:

$$\hat{\mathbf{p}'}^\mathbf{T}\mathbf{T}'^{-\mathbf{T}}\mathbf{F}\mathbf{T}^{-\mathbf{1}}\hat{\mathbf{p}} = \mathbf{0} \tag{3.22}$$

[3] If $\mathbf{A}_r$ denotes the leading $r \times r$ principal sub-matrix of an $n \times n$ symmetric matrix $\mathbf{A}$, and $\lambda_i(\mathbf{A})$ represents the i-th largest eigenvalue of $\mathbf{A}$, then for $r = 1, 2, \ldots, n-1$ the following interlacing property holds [18]:

$$\lambda_{r+1}(\mathbf{A}_{r+1}) \leq \lambda_r(\mathbf{A}_r) \leq \lambda_r(\mathbf{A}_{r+1}) \leq \ldots \leq \lambda_2(\mathbf{A}_{r+1}) \leq \lambda_1(\mathbf{A}_r) \leq \lambda_1(\mathbf{A}_{r+1})$$

This implies that $\hat{\mathbf{F}}$, the fundamental matrix for the normalised point correspondences is:

$$\hat{\mathbf{F}} = \mathbf{T'^{-T}FT^{-1}} \tag{3.23}$$

and we can recover $\mathbf{F}$ by:

$$\mathbf{F} = \mathbf{T'^{T}\hat{F}T} \tag{3.24}$$

Thus we can outline the normalised 8-point algorithm as follows:

1. Given a set of pairs of matched image points $p_i \leftrightarrow p'_i$, apply the normalising transformations $\mathbf{T}$ and $\mathbf{T}'$ to obtain $\mathbf{\hat{p}_i} = \mathbf{Tp_i}$ and $\mathbf{\hat{p}'_i} = \mathbf{T'p'_i}$.
2. Compute the fundamental matrix $\hat{\mathbf{F}}$ corresponding to these normalised points.
3. Recover the fundamental matrix $\mathbf{F}$ corresponding to the 'un-normalised' points as $\mathbf{F} = \mathbf{T'^{T}\hat{F}T}$.

In the sequel we will refer to the normalised implementations as **SVDNORM** and **EIGNORM**.

3.5.5 Quantifying Success

Our overall aim is to obtain the best possible estimate for the fundamental matrix, based on the linear criterion, for later use in our structure recovery system. In doing so, we can evaluate which combination of normalisation and method of solution (**SVD**/**EIG**) provides the best results. The question is, how to quantify 'best'?

The measure we use is based upon the relationship between matched points and their epipolar lines. Specifically, given $\mathbf{l}' = \mathbf{Fp}$, we know that p' should lie somewhere on l'. In fact, this is only exactly true when the computed $\mathbf{F}$ is perfect and in practice the point will actually lie some distance from its epipolar line. We can use this as a measure of the quality of $\mathbf{F}$. Formally, we define the quality of a fundamental matrix, denoted by Q_F, to be the average perpendicular distance of each point from its corresponding epipolar line. Thus we would like Q_F to be as low as possible. A method for computing the perpendicular distance of a homogeneous point from a homogeneous line is given in Appendix A.

The aim of the normalisation process is to obtain a better estimate for $\mathbf{F}$ by improving the conditioning of a linear system. In this respect, its success or failure is easy to quantify. For the **SVD** method we keep track of the condition number, κ, of the matrix $\hat{\mathbf{A}}$ in $\hat{\mathbf{A}}\hat{\mathbf{f}} = -\mathbf{u}'$ (section 3.5.1). For the **EIG** method of solution, we are interested in the condition number of matrix $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ in $\mathbf{A}\mathbf{f} = \mathbf{0}$ (section 3.5.2).

3.6 Results and Conclusions

3.6.1 Initial Experiments

In this section we describe the results of our experiments to determine the effects of the normalising transform on our two implementations of the 8-point algorithm. Experiments were performed on a variety of synthetic data and real images, using manual and automatic feature detection/matching methods. Here we give a representative sample of the results obtained.

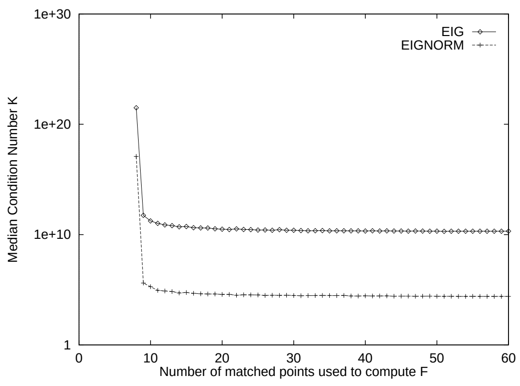
In order to be able to make a valid comparison, our experiments follow along similar lines to those carried out in Hartley's original paper on normalisation [26], in that we show the effect of varying the number of matched points used in the computation of the fundamental matrix. Thus, given a dataset containing M matched points, we begin by selecting a random 8-point subset, S, which is used to obtain an estimate for $\mathbf{F}$. The condition number, κ, of the solved system of equations is recorded, as is the quality measure, Q_F. This process is repeated for a number of *trials*, each time selecting a different random subset of matched points. Once all these trials are completed, we then do exactly the same thing using 9-point subsets, then 10-point subsets, etc., until we have computed $\mathbf{F}$ using all subset sizes $S \in [8 \ldots M]$.

For each dataset we plot graphs which show how κ and Q_F change as S is varied. Here we use the *median* values obtained over all the trials for each size subset. Note that, regardless of the subset size, all M matches are used in the determination of Q_F. In addition, we vary the number of trials used per subset as a fraction of the total number of S-point subsets in M. A pseudo-code overview of the experiments is as follows:

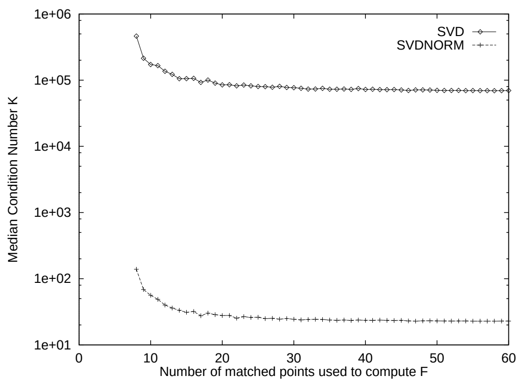
```
Given an M-point dataset:

for subset size S = 8 to M
{
  for some variable number of trials T
  {
    randomly select an S-point subset from M
    compute F using this subset and each of the 4 methods
    enforce the rank constraint
    record the kappa and Qf values for this particular subset
  }
  record the median and best kappa/Qf values for this size subset
}
```

3.6.2 Synthetic Data

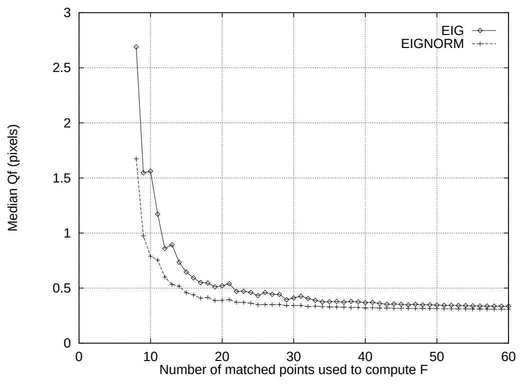


(a)

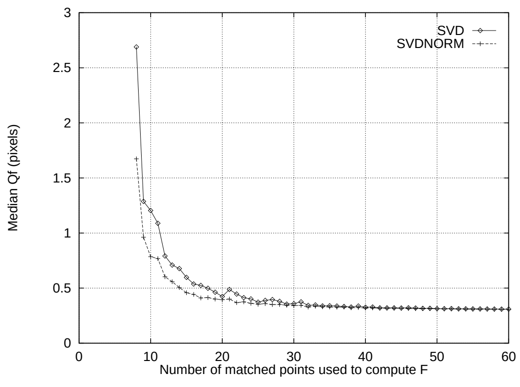


(b)

Figure 3.3: Graphs of S against κ using perfect synthetic data



(a)



(b)

Figure 3.4: Graphs of S against Q_F using perfect synthetic data

We begin by applying the algorithms to a perfect dataset. The idea was to obtain a frame of reference for future experiments and to see whether any differences could be detected when the algorithms were used under ideal conditions. We simulated a scene consisting of 60 randomly generated points within a constrained 3D volume. Two arbitrarily placed cameras project the 3D scene points into the perfect 2D matched data. Figures 3.3 and 3.4 show how varying the number of points used in the fundamental matrix calculation affects Q_F and κ.

Looking at the graphs of κ, two things are immediately obvious. Firstly, it is clear that normalisation has resulted in a dramatic reduction in the condition number for both the **SVDNORM** and **EIGNORM** methods, the improvement being of the order of 10^6 and 10^3 respectively. Secondly, it would seem that as the subset size increases, so κ is reduced. It is also particularly noticeable that the **EIG** implementation, in both its basic and normalised forms, is extremely ill-conditioned when 8-point subsets are used to compute $\mathbf{F}$.

As one would expect for this kind of dataset, the values of Q_F are very low, on the whole to sub-pixel accuracy. However, it does look as though the **EIG** method is producing marginally more accurate results. One thing that is also worth mentioning at this point is that for any given 8-point subset, both methods determine an exact solution for the elements of the fundamental matrix. Thus they generate $\mathbf{F}$ matrices that are the same, to within a scale factor, and hence, give identical values for Q_F.

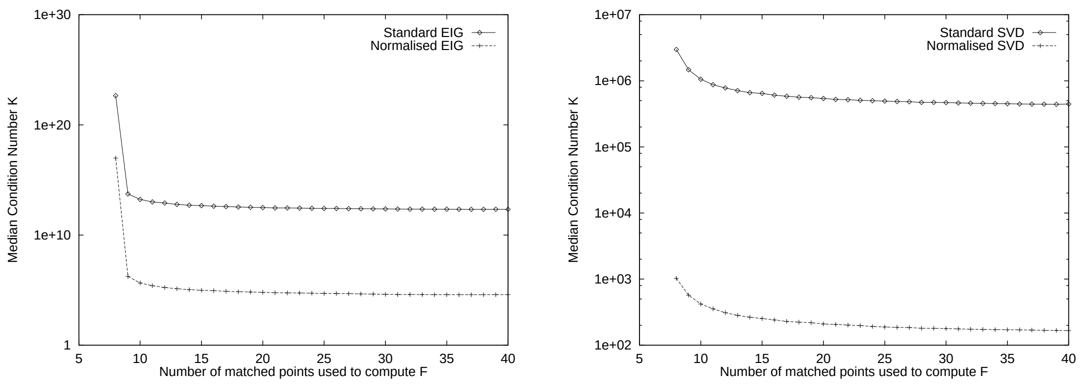
3.6.3 Real Data: Corridor Images

The two images used here are part of a larger, widely-used sequence of a corridor. [4] Feature detection and matching was done by hand, to produce 40 matched point pairs.

Figure 3.5 shows that, as with perfect data, normalisation leads to a reduction in the condition number of the system, for both methods of solution, but, again, we note the high κ values for **EIG** and **EIGNORM** when using exactly eight matched points. However, overall the improvements were of the order of 10^8 for **EIGNORM** and 10^3 for **SVDNORM**.

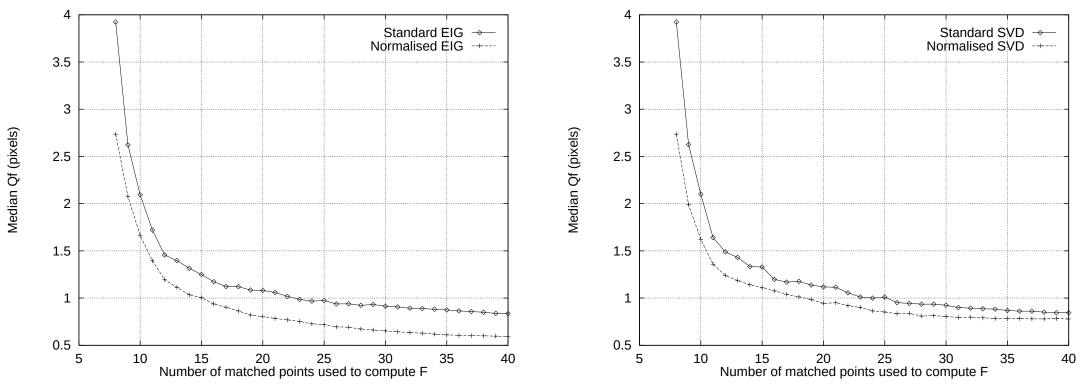
The impact this has on Q_F is easier to see than with the previous dataset (figure 3.6). Both

[4] This sequence was provided by the Robotics Research Group at the University of Oxford.



(a) (b)

Figure 3.5: Graphs of S against κ for the corridor image pair



(a) (b)

Figure 3.6: Graphs of S against Q_F for the corridor image pair

normalised methods result in a small, but consistent, decrease in perpendicular pixel error, relative to their basic counterparts. The sub-pixel accuracy can be attributed to the fact that this manually generated dataset contains only small errors in localisation and no false matches.

Overall, *EIGNORM* is the best performer, but there is very little to choose between any of the four methods. This is emphasised by figure 3.7, which shows the image pair overlaid by epipolar lines generated by the best[5] fundamental matrices, obtained using each of the four methods. As can be seen, the locations of the epipoles and lowest Q_F values are very similar.

3.6.4 Real Data: House Images

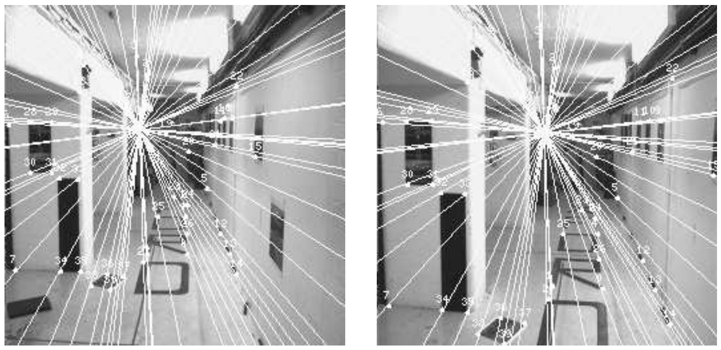
These images of a toy house were obtained from the VASC image database[6]. They are part of a large image sequence (over 180 images) which has been used extensively for machine vision research, particularly structure from motion, for example [48]. The images were processed using the INRIA epipolar geometry server[7], which in this case found a set of 116, rather noisy, point matches (see figure 3.8).

Figure 3.9 shows that, once again, the normalisation process has the desired effect of improving the conditioning of the systems, a reduction in κ of the order of 10^{10} for **EIGNORM** and 10^6 for **SVDNORM**. As before, this leads to a decrease in the median values of Q_F, as shown in figure 3.10. For this noisy dataset, the beneficial effect is more dramatic than we have seen previously, reducing Q_F by more than 3 pixels, when the computation involves large subsets. It is also noticeable that, for the basic **EIG** and **SVD** methods, Q_F begins to increase as the subset size approaches the total number of matched points. This is another feature of our noisy dataset. When large numbers of points are used, there is a greater likelihood that the subset will include false point matches, thus degrading the least-squares solution. However, the monotonically decreas-

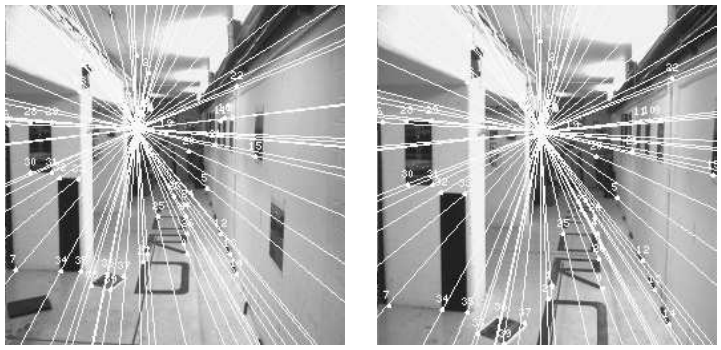
[5] The fundamental matrix with the lowest Q_F over all trials and all subset sizes.

[6] Available on the World Wide Web at http://www.ius.cs.cmu.edu/IUS/ppt_usr0/yx/idbm/image_html

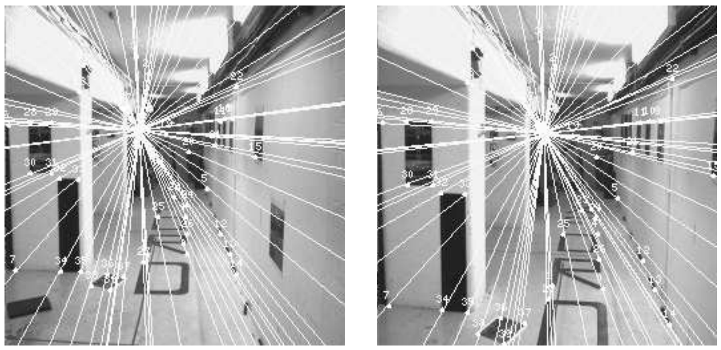
[7] This enables users to have a pair of their own images processed remotely on the INRIA computer, via the World Wide Web. Matched point correspondences are obtained [71] and fed into an algorithm which computes the epipolar geometry [8]. All resulting information is available, for example detected/matched features, the fundamental matrix, images overlaid with matches, epipolar lines etc. This provided an alternative method of obtaining point matches and a useful check on the accuracy of our own results. The URL is: http://www.inria.fr/robotvis/demo/f-http/html/



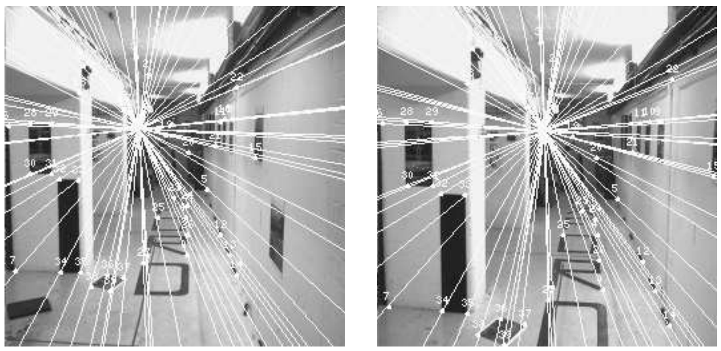
(a) **EIG** $Q_F = 0.54$



(b) **EIGNORM** $Q_F = 0.54$



(c) **SVD** $Q_F = 0.56$



(d) **SVDNORM** $Q_F = 0.55$

Figure 3.7: Sample epipolar lines using the four best $\mathbf{F}$ matrices from the corridor image pair.

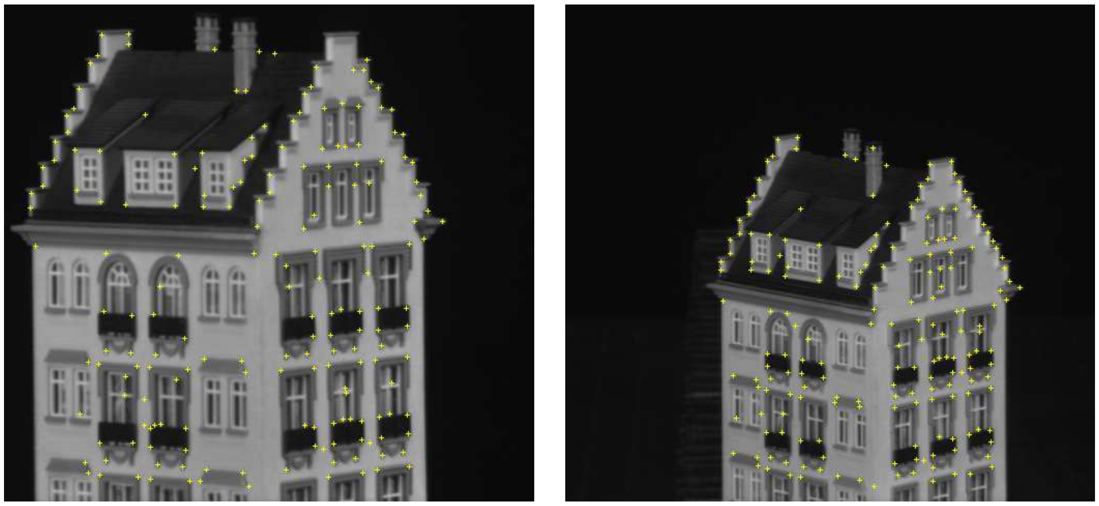
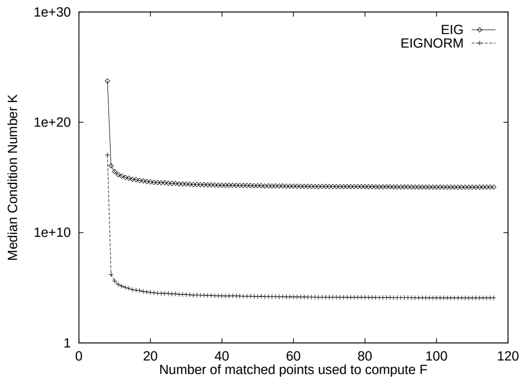
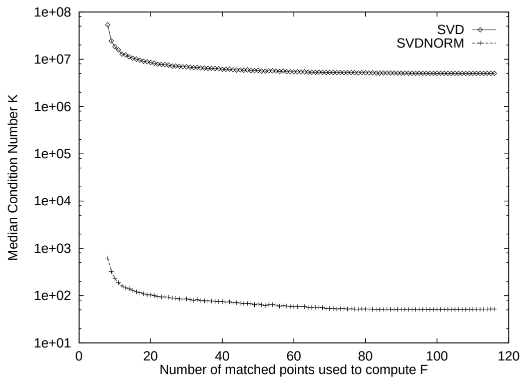


Figure 3.8: Results of the INRIA matching algorithm for the house image pair.

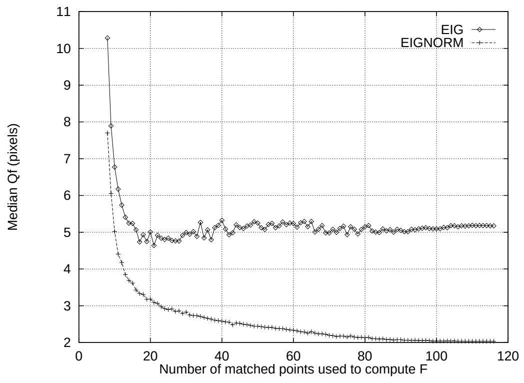


(a)

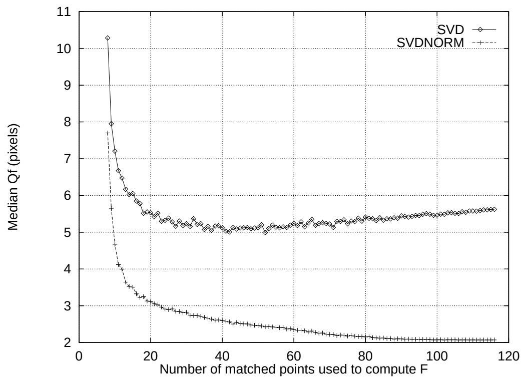


(b)

Figure 3.9: Graphs of S against κ for the house image pair



(a)



(b)

Figure 3.10: Graphs of S against Q_F for the house image pair

ing curves for the **EIGNORM** and **SVDNORM** methods suggest that the normalisation process goes some way towards counteracting this effect.

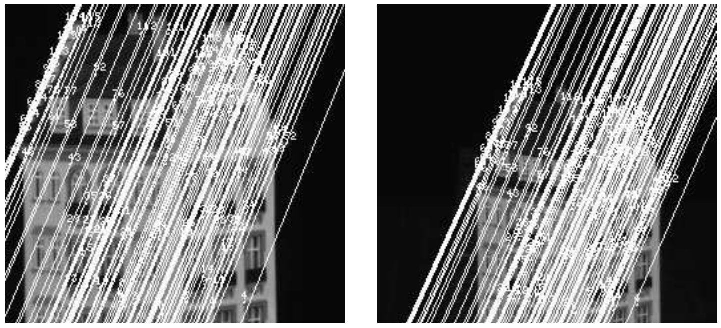
The two normalised methods are clearly the best performers on this dataset, with **EIGNORM** doing slightly better than **SVDNORM**. Although the basic methods give relatively poor results for median Q_F values, figure 3.11 shows that the best fundamental matrices obtained over all their trials are very similar to those of their normalised counterparts.

3.7 Initial Conclusions

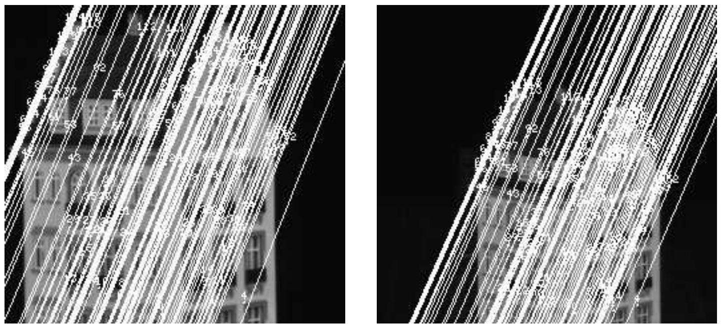
In every experiment we performed, the normalisation process had the desired effect of improving the conditioning of the system of linear equations. For the least eigenvector method, the reduction in κ was of the same order of magnitude as reported by Hartley, at approximately 10^8. The theory behind normalisation is specifically linked to this method of solution and one of the things we were interested in was whether it could also be used to enhance the conditioning of the **SVD** method. Indications are that this is the case, with our results showing a reduction of about 10^4. The magnitude of the improvement is clearly dependent on the quality of the dataset, the most beneficial effects occurring with noisy data, containing may false matches and localisation errors. The same applies with respect to the perpendicular pixel error; normalisation gave the biggest decreases in Q_F on the noisiest datasets.

In every one of our **EIGNORM** experiments the enhanced conditioning resulted in lower values for Q_F. In the vast majority of cases, **SVDNORM** gave only slightly poorer results, but occasionally it was unstable (see figure 3.12).

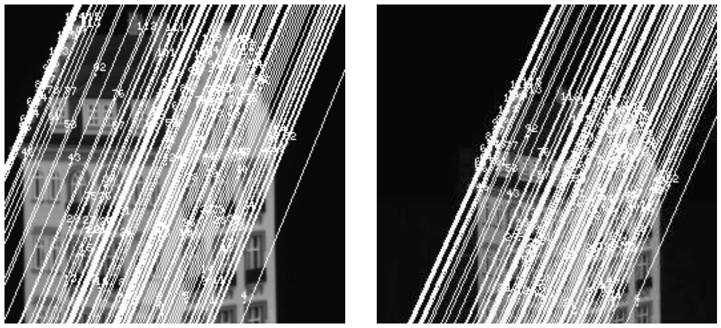
Here, in spite of the lower condition number, **SVDNORM** actually gives worse results for Q_F than basic **SVD**. While the **SVD** method itself gives quite erratic values, it at least shows the familiar decreasing curve as the subset size increases, which is not the case for **SVDNORM**. One possible cause of these anomalies could be the composition of the dataset, which in this case contained a large proportion of projections of planar 3D points. Hence, many of the random matched point subsets would have contained degenerate configurations [62]. We have not yet carried out a more in depth investigation of this phenomenon, since we are primarily interested in a high-level



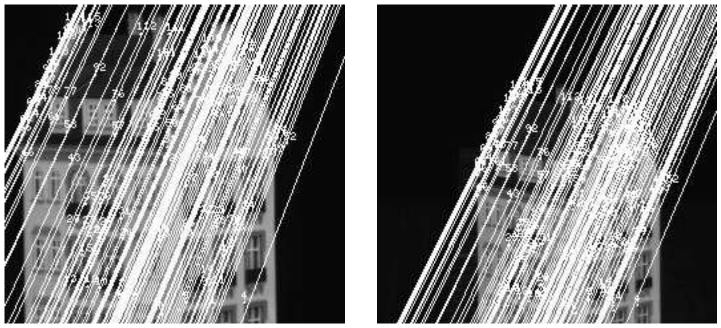
(a) **EIG** $Q_F = 1.83$



(b) **EIGNORM** $Q_F = 1.81$



(c) **SVD** $Q_F = 1.82$



(d) **SVDNORM** $Q_F = 1.82$

Figure 3.11: Sample epipolar lines using the four best $\mathbf{F}$ matrices from the house image pair.

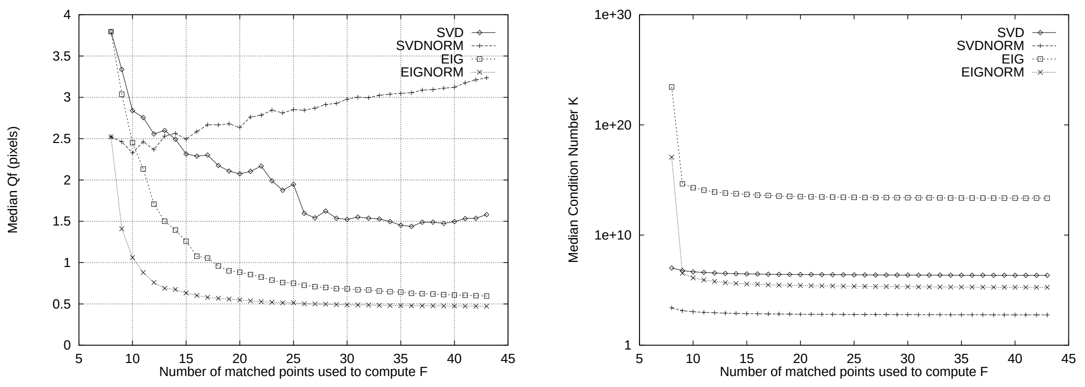


Figure 3.12: Instability of the **SVDNORM** method

comparison of the methods of solution. Here we merely note its, albeit infrequent, occurrence and the fact that the **EIG** and **EIGNORM** methods seem unaffected.

In short, for this series of tests, the **EIGNORM** method produced consistently lower perpendicular pixel errors and better estimates for the coordinates of the epipoles. The algorithm copes very well with noisy data, and its performance degrades gracefully as the level of noise is increased. Therefore, we have concluded that of the various implementations tested, the normalised eigensystem method is the clear winner, and from now on this is the method of implementation we have used.

3.7.1 Further Experiments

The experiments we have performed so far have confirmed Hartley's findings, that the normalisation process does lead to more accurate estimation of the fundamental matrix via the 8-point algorithm. However, the way these experiments were carried out is not a very practicable solution to the problem. Rather than doing a laborious series of trials for different sized subsets we would prefer a more efficient way of finding the best fundamental matrix for a given dataset. Looking at the graphs of subset size against perpendicular pixel error that we have presented, one might conclude that the best way to calculate **F** would be to use the largest possible subset size. However, it must be remembered that these graphs plot the *median* values of Q_F. Consider instead, figure 3.13. This is an example plot of the *lowest* values of Q_F for each subset size, obtained during the experiment on the house images described previously, but it is typical of the results for other datasets.

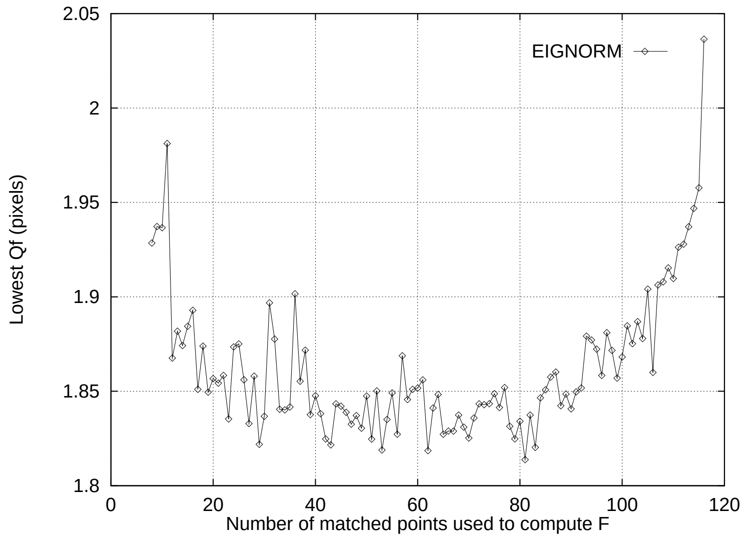


Figure 3.13: Example Lowest Q_F Values.

Looked at in isolation, without the swamping effect of being compared with the high error, basic method, it is clear that the **EIGNORM** performance also degrades as the subset size approaches the total number of matches. As has already been mentioned, this is due to the fact that large subsets are more likely to include outliers in the data; false matches which taint the least-squares result. The use of large subsets when computing **F** causes other problems. Clearly, as the size of our subset approaches that of the complete matched point set $(S \rightarrow M)$, the number of permutations of points in that subset increases exponentially, as does the number of trials required to find the best S-point subset. In addition larger subsets incur a computational overhead, because they lead to bigger systems of equations which take longer to solve.

However, it is equally apparent from this, and many other experiments on real data, that the answer is not to do the computation using the smallest possible (8-point) subset. It may well be the case that a fundamental matrix estimated from an 8-point subset is very accurate in terms of those 8 points, but is not so good when applied to all the matched points. It seems that the optimum subset size i.e. the one that gives the lowest Q_F, can be anywhere except at the extremes of the range $S \in [8 \ldots M]$. The question is, how to find it? A brute force and ignorance solution, which we will refer to as **BEST-F**, is simply to choose the best **F** matrix over all trials and subset sizes. However, we would prefer a more intelligent and efficient approach.

The solution we employ is a variant of the RANSAC parameter estimation paradigm, which we denote by **RANSAC***.[8] As before a series of trials are performed using randomly selected subsets of the total number of matched points, but here 8-point subsets are chosen. At each trial, we take the computed **F** and find all the point matches whose perpendicular pixel error is beneath some threshold value. These matches form the *consensus set*. The (≥ 8) points in the consensus set are then used to compute another estimate for **F** for which we calculate Q_F, in the usual manner. The old consensus set is replaced if the new consensus set is of equal or larger size and has a lower Q_F. The process terminates either once all trials have been completed or the consensus set size or Q_F reach a specified target. In detail, the **RANSAC*** version of our experiments is as follows:

Given a dataset, M, of point matches, then for some fixed number of trials:

1. Randomly select an 8-point subset from M.
2. Compute **F** for this subset using **EIGNORM**.
3. Enforce the rank constraint.
4. Determine the consensus set C of matches whose perpendicular pixel error is within some threshold.
5. Compute a new $\mathbf{F}'$ based on C.
6. Calculate Q_F for $\mathbf{F}'$.
7. Replace the old consensus set if C is the same size or bigger, and has lower Q_F value.
8. Terminate if the size of C or Q_F reach specified targets.

At the end of this procedure, we hope to have obtained a fundamental matrix, based on many of the matched points, with a low Q_F value, and to have done so in a relatively small number of trials.

[8] For a full description, refer to appendix B.

3.7.2 Further Results

There is little point in attempting to quantify the increase in speed of **RANSAC*** relative to the **BEST-F** approach, since, by altering the number of trials per subset size in the latter case, we could come up with whatever figures we liked. Instead, we note that for a given dataset, however many trials per subset size are needed in order to obtain a good estimate for **F** using **BEST-F**, invariably, **RANSAC*** requires fewer trials to give a fundamental matrix which is almost as accurate. Some example results on different datasets[9] are given in table 3.1.

Dataset	Matched	**BEST-F**			**RANSAC***		
	Points	Q_F	Subset Size	Trials	Q_F	Consensus Size	Trials
Perfect	60	0.30	52	4413	0.31	45	2340
Corridor	40	0.54	31	23674	0.62	23	6166
Quadrangle	43	0.43	23	32842	0.55	32	4570
House	116	1.81	50	42839	1.90	43	5666

Table 3.1: Comparison of results of obtaining **F** using the **BEST-F** and **RANSAC*** algorithms

The **RANSAC*** method results in slightly inferior Q_F values than can be obtained using **BEST-F**. However, this small decrease in accuracy is outweighed by big improvements in efficiency. To put the performance of **RANSAC*** into perspective, table 3.2 shows how it compares against some of the more complicated alternative methods for estimating the fundamental matrix.

NONLIN is a non-linear least-squares method, **LMEDS** is the least *median* of squares and **M-EST** uses the M-estimators technique, as discussed in Chapter 2. Clearly, there is very little to choose between the various implementations. With perfect data, as one would expect, they all give virtually the same results, especially when applied to our noisiest dataset, the house images. There is more of a differential when using hand-matched data, containing only small localisation errors, but we are still only talking in terms of a few hundredths of a pixel accuracy. The performance of **RANSAC*** is even more impressive with our automatically processed house dataset, which

[9]We have already seen the perfect dataset and the corridor and house images. The quadrangle dataset is a pair of hand-matched images from an outdoor image sequence of part of the Leeds University campus. The entire image sequence is shown in Appendix C.

Dataset	Q_F			
	RANSAC*	**NONLIN**	**LMEDS**	**M-EST**
Perfect	0.31	0.31	0.31	0.31
Corridor	0.62	0.56	0.56	0.55
Quadrangle	0.55	0.49	0.47	0.45
House	1.90	1.92	1.84	1.90

Table 3.2: Comparing **RANSAC*** with more complicated algorithms

includes many mismatches. Here it is only bettered by the least-median of squares algorithm.

3.8 Conclusions

In this chapter we have investigated Hartley's normalisation technique for improving the performance of the 8-point algorithm. Although this technique is based on numerical analysis of one particular method of solution (**EIG**), we have shown that its beneficial effects are quite general in that its application to another solution method (**SVD**) also leads to more accurate results.

We have demonstrated that, when using the normalised 8-point algorithm (**EIGNORM**) to estimate the fundamental matrix, the size of the matched point subset used in the computation is of great importance. We can obtain an excellent solution simply by carrying out thousands of trials for each possible subset size and then picking the best solution overall, however, this is very time-consuming. To this end, we have embedded **EIGNORM** within a RANSAC-style procedure which attempts to obtain a quick, accurate estimate for the fundamental matrix, based on a medium-sized point subset (**RANSAC***). We do not claim that this will produce an optimum solution, but have shown that results of our simple, linear method are similar to those obtained with other, more complex alternatives. In the next chapter we will use the fundamental matrices acquired using **RANSAC***, as the starting point for our 3D scene reconstruction.

Chapter 4

Reconstruction

4.1 Introduction

In his seminal paper [12] Faugeras showed that it is possible to obtain the projective structure of a scene, given just a set of matched points in a pair of images taken with uncalibrated cameras. This generated a great deal of interest in the development of algorithms for performing *uncalibrated stereo*, that is, recovering the three-dimensional structure of a scene, without explicit knowledge of the intrinsic or extrinsic camera parameters. Descriptions of some of these algorithms have been given in Chapter 2. In the absence of any other information or simplifying assumptions, the reconstruction will be correct up to a projective ambiguity i.e. it will differ from the real Euclidean structure by a projective transformation.

Projective structure can be very useful in its own right, for example, objects may be recognised via projective invariants [56]. However, although geometrically related, it is unlikely that the projective and Euclidean structures will *look* anything like each other. If the intention is to use the recovered structure for graphical reconstruction and subsequent viewing by a person, this clearly poses a problem.

A method of converting from a projective to Euclidean reconstruction was presented in [25]. This uses the constraint of unchanging intrinsic camera parameters, which is the basis of camera self-calibration theory[13], requiring at least three views of the scene, taken with the same camera. The method is highly complex and relies heavily on the Levenberg-Marquardt iterative parameter estimation algorithm [50]. Although it appears to give good results using synthetic data, the method's performance on images of real scenes is less impressive. For this reason, we have adopted a simpler approach, using *ground truth* points. If the Euclidean positions of five or more of the points in the projective structure are known, it is possible to compute the projectivity that maps between them. This can then be applied to *all* the projective points to achieve a full Euclidean reconstruction.

In this chapter we describe our implementation of a Euclidean reconstruction method, based on the properties of the fundamental matrix. Rather than restricting ourselves to a single pair of images, we present a novel incremental version of the basic algorithm which allows us to recover the Euclidean structure of an extended environment viewed over a long image sequence.

4.2 Outline of the method

Before embarking on a detailed description of each of the stages in our implementation, we first provide a brief overview of the entire system. A diagrammatic representation is given in figure 4.1.

We begin by estimating the fundamental matrix for the first pair of images in the sequence, using the algorithm described in the previous chapter. This is then factorised in a manner which allows us to construct estimates for the pair of camera matrices used in the formation of the two images. Next, we recover the projective structure of each of the matched points by computing the intersection of the two rays, projecting back from the two cameras through the corresponding image features.

For the first image pair only, we transform our projective structure into a Euclidean frame, using a projectivity derived from a small number of user-supplied ground truth points. With subsequent images, we proceed in a similar manner. A new fundamental matrix is estimated, using matched points in the current and previous images. As before, this is factorised to obtain a pair of camera matrices and hence, another batch of projective structure. Once again, we upgrade the projective structure using a projective transform, but here, the ground truth information is not

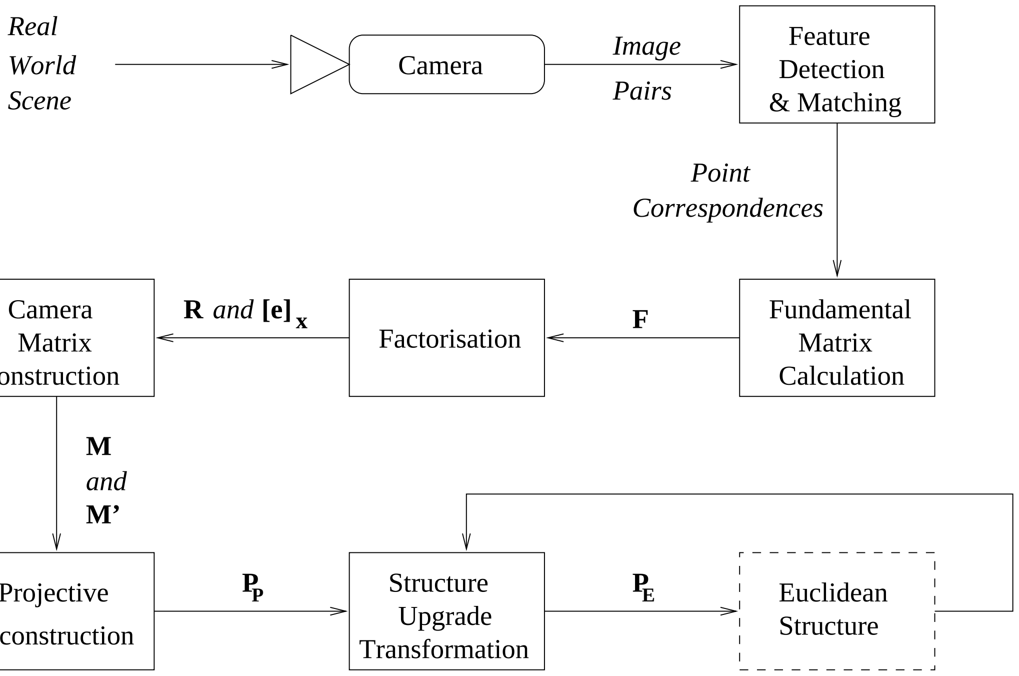

Real World Scene
Camera
Image Pairs
Feature Detection & Matching
Point Correspondences
Fundamental Matrix Calculation
F
Factorisation
R and [e]x
Camera Matrix Construction
M and M'
Projective Reconstruction
P_P
Structure Upgrade Transformation
P_E
Euclidean Structure

Figure 4.1: Overview of the system

supplied directly by the user. Instead, we use common points from our previously computed Euclidean structure.

The entire process is repeated, generating new Euclidean structure as we step through each pair of images in the sequence. Note that, as the sequence is processed, old points will disappear and new ones come into view: there is no requirement for points to be visible in all the images.

4.3 Image Sequence Processing

Before embarking on a description of the reconstruction algorithm we first consider some of the inherent problems of image sequence processing and the effects these have on the overall design of our system.

Any system which involves the processing of long image sequences must address a number of issues. First of all, unless the system is capable of processing live video directly from a camera, or some other input source such as a VCR, then the sequence must be digitised onto disk, and thus we need to look at the space requirements. Take, for example, our quadrangle sequence. In this case, six and a half minutes of original video footage, digitised at full frame rate (25 frames per second), took up over a gigabyte of disk space! We simply did not have enough resources to keep this amount of data on our system for any length of time, so had to find a way to reduce the size.

One obvious solution was simply to reduce the digitising frame rate. When attempting to do so, it was important to remember that each pair of images in the resulting sequence must contain enough overlap to allow us to obtain a set of matched features, for use in our fundamental matrix algorithm. Now, our footage was filmed by walking around with an ordinary palm-corder and the route we followed involved a number of twists and turns, some of them quite sharp. As such, it was extremely difficult to keep the camera moving at a steady pace and so some parts of the scene passed in and out of view much more quickly than others. Where this was so, it was essential to digitise at full frame rate, to ensure enough inter-frame overlap, but elsewhere this resulted in lots of almost identical, redundant frames.

Ideally, we would have liked to have been able to vary the digitisation rate, based on the speed of camera motion. Without this facility, the best solution we could manage was to digitise at full frame rate and then manually extract representative frames, discarding the rest. This process reduced over 10000 digitised frames to 104, resulting in the image sequence shown in Appendix C, which only takes up 10 megabytes of disk space.

Unfortunately, this space saving comes at a cost. Suppose we have processed a pair of images from our sequence, with a corner detector, for example [22, 57, 67]. The next task is to solve the correspondence problem i.e. find the set of matching corners between the two images. The fact that there is such a relatively large baseline between our images, causes problems for *local* strategies, such as nearest-neighbour, which search for a match in the area of the second image corresponding to the location of the other corner in the first. For example, consider figure 4.2.

This shows the result of applying a simple local matching algorithm, based on cross-correlation, to a set of corner features, found via the Plessey detector [22], in a pair of the corridor sequence images. The usual match search area parameter of about ten pixels is of no use for this image pair, as the disparity is much greater than that. However, it can be seen that using a larger value is not the answer. The image on the right is overlaid with matched corners, in red, and yellow lines

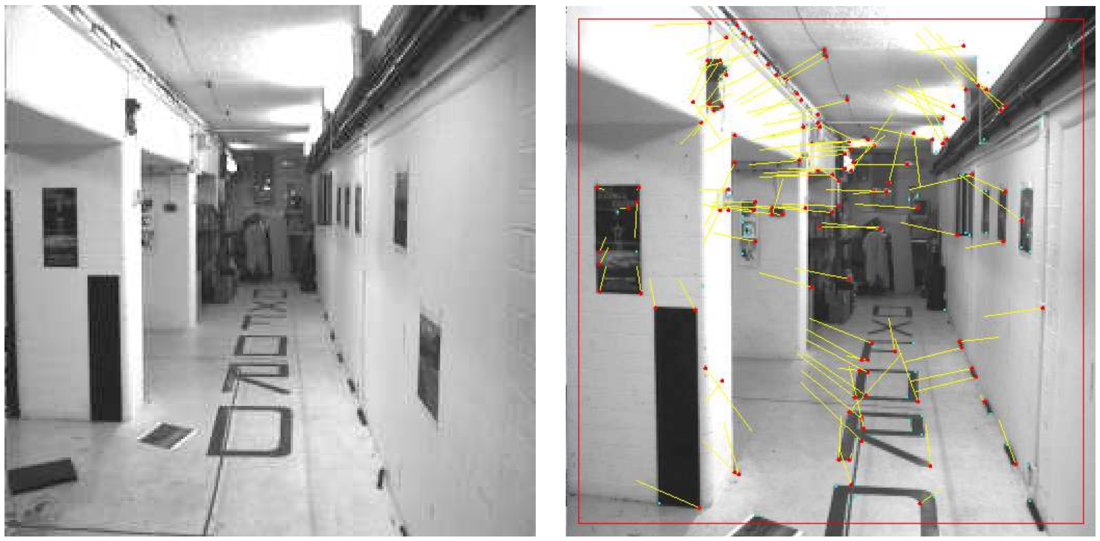


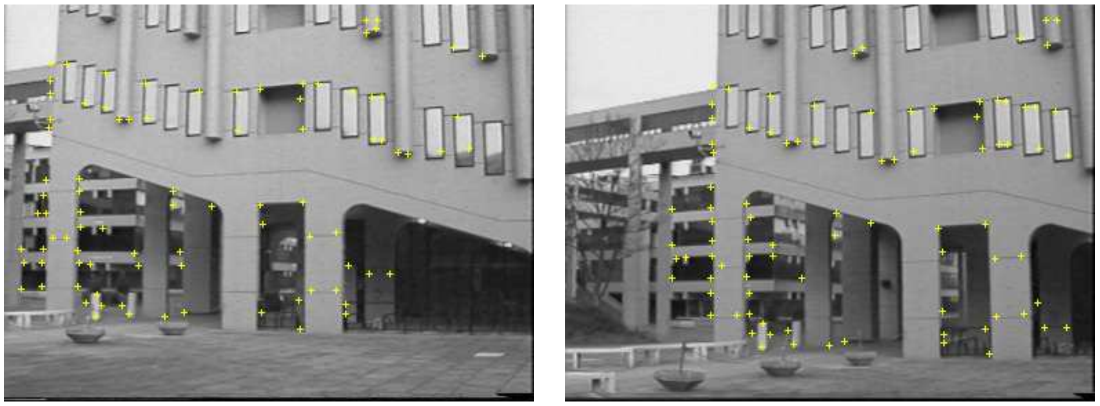
Figure 4.2: Poor performance of a local matching algorithm.

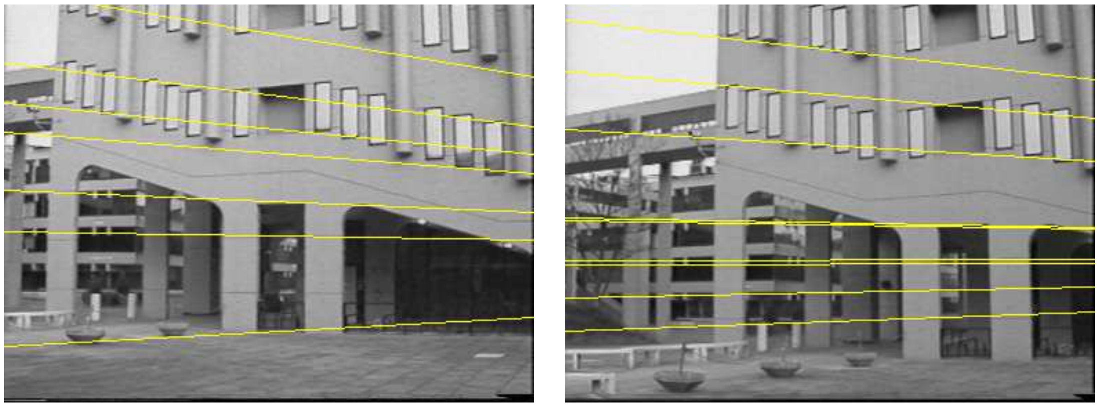
indicating the corner trajectories. Clearly the matching process has not been very successful.

In order to have a chance of finding the correct match, the search area must be increased. However, this not only increases the computational cost of the process, it also makes it more prone to false matches. This is particularly so for scenes containing repeated structure (windows etc), where it is easy to match one instance incorrectly against another.

The effect of false matches can be seen in Figure 4.3. Figure 4.3(a) shows the results of processing the first pair of quadrangle images with the INRIA point detection and matching algorithm [71]. Evidently, a significant number of false matches have been detected. The fundamental matrix estimated from these noisy matches does not encapsulate the correct epipolar geometry for this pair of images. This can be seen by comparing the epipolar lines in figure 4.3(a), with those in 4.3(c), which were generated by a fundamental matrix computed from hand-matched data.

We have already mentioned one possible solution in the previous chapter. The fundamental matrix constrains the location of a matched point to lie on a given epipolar line, which can reduce the search space dramatically. Unfortunately, this is like the ‘chicken and the egg’ situation, in that we need to find at least eight point matches in order to compute the fundamental matrix! However, if an initial set of matches is obtained, by some other means, the fundamental matrix approach can then be used to discard false matches in the set, as described in [71]. A variation on this theme is


(a) Noisy matched data.


(b) Epipolar lines from noisy matches.

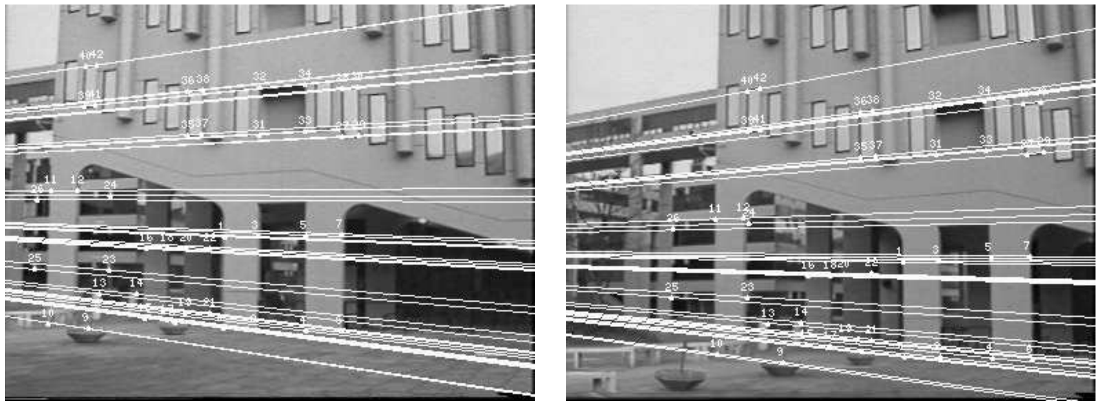

(c) Epipolar lines from hand-matched data..

Figure 4.3: Effects of false matches on epipolar geometry

discussed in [4], which matches over three images using the trifocal tensor.

It was not the objective of this work to investigate automatic feature matching, which is a research topic in itself. For this reason, a simple, manual feature detection and matching system was developed, using an X11-based graphical user interface (see figure 4.4). The user can process each image of the sequence in turn, matching existing features or specifying new ones, by clicking the mouse button at the desired image location. Matched features are highlighted in red, unmatched features are either old (yellow) or new (orange).

4.4 Constructing Camera Matrices

In the previous chapter a method was described for estimating the fundamental matrix, $\mathbf{F}$, from a pair of views of a scene, taken with an uncalibrated camera. Acquiring $\mathbf{F}$ is the key to recovering the projective structure of the scene, since it encapsulates all the geometric information relating the scene and the cameras that can be extracted from a set of matched image features. In particular, knowing $\mathbf{F}$ allows us to construct camera matrices for our two cameras.

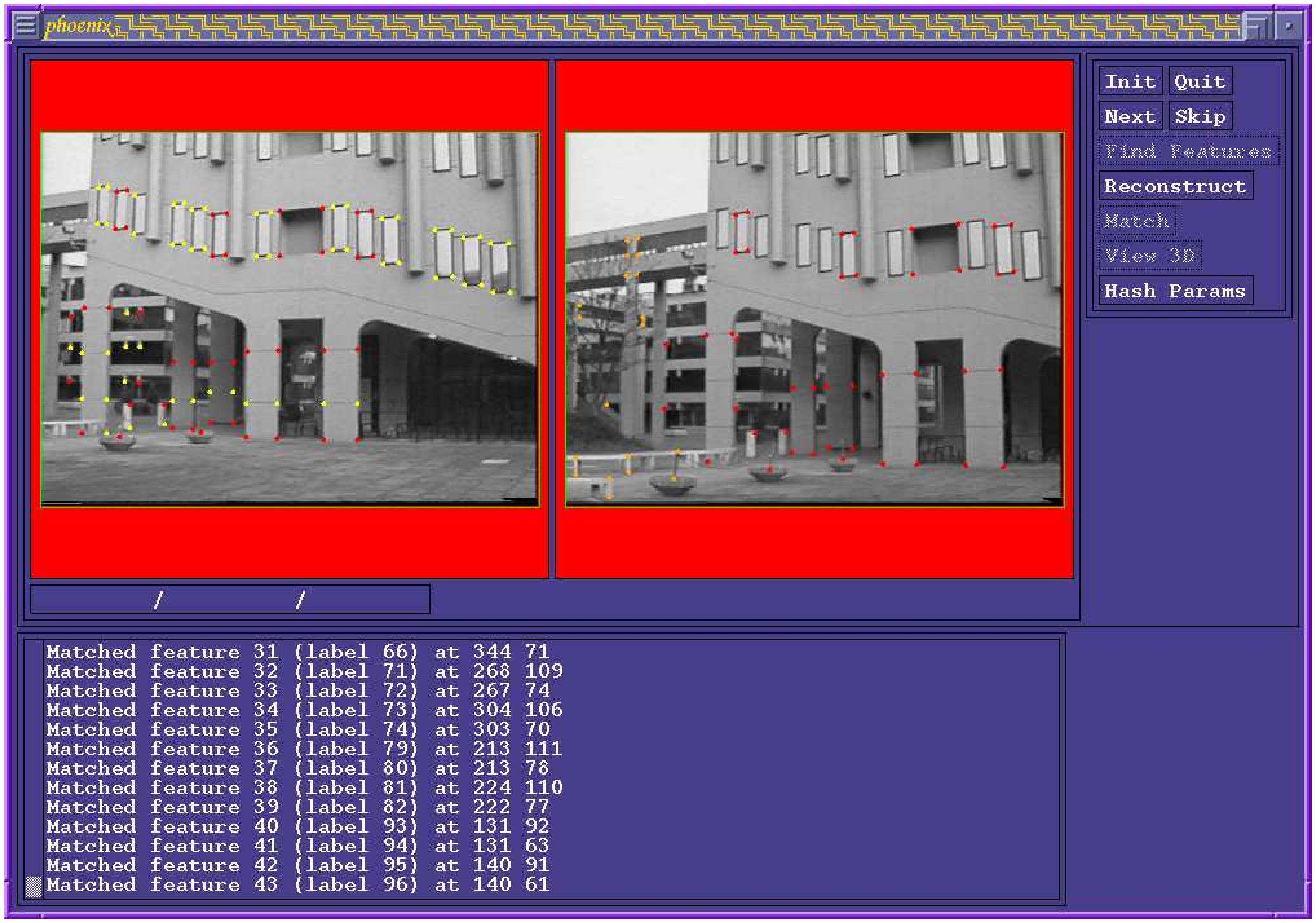
The reason for this is directly related to the derivation of the linear criterion in section 3.3. There it was shown that, by choosing $\mathbf{C} = (\mathbf{I}|\mathbf{0})$ and $\mathbf{C}' = (\mathbf{R}| - \mathbf{Rt})$ as our partitioned camera matrices, the fundamental matrix could be factorised as $\mathbf{F} = \mathbf{RS}$, where $\mathbf{S} = [\mathbf{t}]_\times$. Now consider the more general case, where the two camera matrices are $\mathbf{C} = (\mathbf{R}|-\mathbf{Rt})$ and $\mathbf{C}' = (\mathbf{R}'|-\mathbf{R}'\mathbf{t}')$. As before, it is possible to determine the epipolar line corresponding to a point $(u, v, w)^T$ in the first image. Two points which must lie on this line are the images under $\mathbf{C}'$ of the first camera's optical centre and the point at infinity, given by

$$\begin{pmatrix} t \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{R}^{-1}(u, v, w) \\ 0 \end{pmatrix} \tag{4.1}$$

These project to $\mathbf{R}'(\mathbf{t} - \mathbf{t}')$ and $\mathbf{R}'\mathbf{R}^{-1}(u, v, w)$ in the second image. The epipolar line, $\mathbf{l}' = (q, r, s)$, through these points is given by the cross product

$$(q, r, s) = \mathbf{R}'(\mathbf{t} - \mathbf{t}') \times \mathbf{R}'\mathbf{R}^{-1}(u, v, w) \tag{4.2}$$

Figure 4.4: Our manual feature detection and matching user interface.



```
Matched feature 31 (label 66) at 344 71
Matched feature 32 (label 71) at 268 109
Matched feature 33 (label 72) at 267 74
Matched feature 34 (label 73) at 304 106
Matched feature 35 (label 74) at 303 70
Matched feature 36 (label 79) at 213 111
Matched feature 37 (label 80) at 213 78
Matched feature 38 (label 81) at 224 110
Matched feature 39 (label 82) at 222 77
Matched feature 40 (label 93) at 131 92
Matched feature 41 (label 94) at 131 63
Matched feature 42 (label 95) at 140 91
Matched feature 43 (label 96) at 140 61
```

Which, by the properties of cofactor matrices, can be rewritten

$$(q, r, s) = \mathbf{R}'^{*}((\mathbf{t} - \mathbf{t}') \times \mathbf{R}^{-1}(u, v, w)) \tag{4.3}$$

and

$$(q, r, s) = \underbrace{\mathbf{R}'^{*}[\mathbf{t} - \mathbf{t}']_{\times}\mathbf{R}^{-1}}_{\mathbf{F}}(u, v, w) \tag{4.4}$$

and thus we obtain an expression for $\mathbf{F}$ in terms of the components of our camera matrices.

$$\mathbf{F} = \mathbf{R}'^{*}[\mathbf{t} - \mathbf{t}']_{\times}\mathbf{R}^{-1} \tag{4.5}$$

It has been shown [23] that this relationship does not determine the two camera matrices uniquely. In particular, if $\mathbf{C_1}$ and $\mathbf{C_1'}$ are two camera transforms satisfying equation 4.5, then so are $\mathbf{C_2} = \mathbf{HC_1}$ and $\mathbf{C_2'} = \mathbf{HC_1'}$, where the matrix $\mathbf{H}$ is a 4×4 projectivity. Consequently, the 3D structure of a scene derived from any such pair of camera matrices is defined only up to an arbitrary projective transformation. The task now is to construct a pair of camera matrices for which equation 4.5 holds. A number of different ways of doing this have been proposed, for example [23, 24, 4, 53]. A hybrid of some of these methods is described, which we believe leads to a more intuitive solution.

4.5 Factorising F

We follow convention and choose the origin of the world coordinate system as the optical centre of the first camera, its axes aligned with the camera axes. The second camera is displaced from the first by some translation and rotation, giving the two familiar camera matrices, $\mathbf{C} = (\mathbf{I}|\mathbf{0})$ and $\mathbf{C}' = (\mathbf{R}| - \mathbf{R}'\mathbf{t}')$. Substituting into equation 4.5 gives:

$$\mathbf{F} = \mathbf{R}'^{*}[-\mathbf{t}']_{\times} \tag{4.6}$$

The problem now is to find a way to factorise $\mathbf{F}$ in the above form.

The first step is to compute the singular value decomposition $\mathbf{F} = \mathbf{UDV}^T$, where $\mathbf{U}$ and $\mathbf{V}^T$ are orthogonal matrices and $\mathbf{D}$ is the diagonal matrix $(r, s, 0)$. [1] Note that one of the singular

[1] Some swapping of matrix rows and columns may be necessary so that $r > s > 0$.

values will always be zero, due to the enforcement of the rank constraint, discussed in section 3.5.3. Defining the two matrices:

$$\mathbf{E} = \begin{pmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{pmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{4.7}$$

The fundamental matrix can now be factorised as:

$$\mathbf{F} = \mathrm{M}^*[\mathbf{e}]_\times \tag{4.8}$$

where

$$\mathrm{M}^* = \mathbf{U}diag(r, s, \gamma)\mathbf{E}\mathbf{V}^T \tag{4.9}$$

with γ any non-zero number, best chosen to lie between r and s, to make M as well-conditioned as possible[2], and

$$[\mathbf{e}]_\times = \mathbf{V}\mathbf{Z}\mathbf{V}^T \tag{4.10}$$

where $\mathbf{e}$ is the coordinate vector for the epipole in the first image.

This works because the process of singular value decomposition explicitly constructs orthonormal bases for the nullspace and range of a matrix. Specifically, the rows of $\mathbf{V}^T$ (denoted $\mathbf{V}_i^T$ for $i = 1, 2, 3$) whose corresponding singular values are zero are an orthonormal basis for the nullspace. Given the ordering of the singular values and the fact that $\mathbf{Fe} = \mathbf{0}$, it is clear that $\mathbf{V}_1^T \times \mathbf{V}_2^T = \mathbf{V}_3^T = \mathbf{e}$, which accounts for $[\mathbf{e}]_\times = \mathbf{V}\mathbf{Z}\mathbf{V}^T$. Furthermore, since $\mathbf{F} = \mathrm{M}^*[\mathbf{e}]_\times$, it follows that $\mathbf{e}^T\mathrm{M}^T\mathbf{F} \approx \mathbf{e}^T\mathrm{M}^T\mathrm{M}^*[\mathbf{e}]_\times = \mathbf{e}^T[\mathbf{e}]_\times = 0$. Hence, $\mathbf{F}^T(\mathrm{M}\mathbf{e}) = 0$ and so $\mathrm{M}\mathbf{e} \approx \mathbf{e}'$.

The end result of this factorisation, is the following pair of camera matrices

$$\mathbf{C} = (\mathbf{I}|0) \quad \text{and} \quad \mathbf{C}' = (\mathrm{M}|\mathbf{p}') \tag{4.11}$$

Once again, it must be stressed that these are in no way intended to be the true camera matrices, but they are related to them by a projective transformation, which is enough to allow us to recover projective structure.

One final point of note with which to end this section: the matrix M^* is the epipolar transform discussed in section 3.2. It provides the mapping between epipolar lines in the two images.

[2]We set $\gamma = (r + s)/2$.

To see this, let p be a point in the first image and let l' be the corresponding epipolar line in the second image, hence $\mathbf{l}' = \mathbf{F}\mathbf{p}$. The epipolar line through p in the first image is $\mathbf{l} = \mathbf{p} \times \mathbf{e} = [\mathbf{e}]_\times \mathbf{p}$. Hence, $\mathrm{M}^*\mathbf{l} = \mathrm{M}^*[\mathbf{e}]_\times \mathbf{p} = \mathbf{F}\mathbf{p} = \mathbf{l}'$, as required. Thus the matrix M maps epipoles to epipoles and the matrix M^* maps epipolar lines to epipolar lines.

4.6 Projective Reconstruction

Suppose $\mathbf{C}$ and $\mathbf{C}'$ are camera matrices, consistent with the fundamental matrix obtained from a set of point matches in a pair of images. Armed with this information, it is a relatively straightforward process to recover the corresponding projective structure of the scene. Consider one such point match $\mathbf{p} = (u, v, 1) \Leftrightarrow \mathbf{p}' = (u', v', 1)$. The point $\mathbf{P}_P = (X, Y, Z, T)$ in $\mathcal{P}^3$ that projects onto p and p' is located at the intersection of the two rays which originate from the optical centres of the cameras and pass through the matched points. This places constraints on $\mathbf{P}_P$, based on the standard perspective projection equations:

$$\begin{pmatrix} \alpha u \\ \alpha v \\ \alpha \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} \tag{4.12}$$

and

$$\begin{pmatrix} \beta u' \\ \beta v' \\ \beta \end{pmatrix} = \begin{pmatrix} C'_{11} & C'_{12} & C'_{13} & C'_{14} \\ C'_{21} & C'_{22} & C'_{23} & C'_{24} \\ C'_{31} & C'_{32} & C'_{33} & C'_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} \tag{4.13}$$

Multiplying out we get

$$\begin{aligned} \alpha u &= C_{11}X + C_{12}Y + C_{13}Z + C_{14}T \\ \alpha v &= C_{21}X + C_{22}Y + C_{23}Z + C_{24}T \\ \alpha &= C_{31}X + C_{32}Y + C_{33}Z + C_{34}T \end{aligned} \tag{4.14}$$

and

$$\begin{aligned} \beta u' &= C'_{11}X + C'_{12}Y + C'_{13}Z + C'_{14}T \\ \beta v' &= C'_{21}X + C'_{22}Y + C'_{23}Z + C'_{24}T \\ \beta &= C'_{31}X + C'_{32}Y + C'_{33}Z + C'_{34}T \end{aligned} \qquad (4.15)$$

All the various camera matrix elements are known, as are u, v and u', v', so, cancelling out the scale factors, α and β, then rearranging, gives a set of four linear equations in the four unknowns X, Y, Z and T.

$$\begin{pmatrix} C_{11} - uC_{31} & C_{12} - uC_{32} & C_{13} - uC_{33} & C_{14} - uC_{34} \\ C_{21} - vC_{31} & C_{22} - vC_{32} & C_{23} - vC_{33} & C_{24} - vC_{34} \\ C'_{11} - u'C'_{31} & C'_{12} - u'C'_{32} & C'_{13} - u'C'_{33} & C'_{14} - u'C'_{34} \\ C'_{21} - v'C'_{31} & C'_{22} - v'C'_{32} & C'_{23} - v'C'_{33} & C'_{24} - v'C'_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (4.16)$$

With perfect data, the solution to this homogeneous system of equations is P_P, the point of intersection of the two rays. In practice, measurement errors mean the rays will not intersect and we solve for the point of their closest approach. We generate and solve such a system for each pair of matched points to build a complete projective reconstruction.

A variation on the above is to constrain the form of the points in $\mathcal{P}^3$ to $\mathbf{P}_P = (X, Y, Z, 1)$. This leads to a system of four linear equations in three unknowns, which can be solved by standard least-squares techniques. However, it has been pointed out [52] that this formulation makes the, possibly invalid, assumption that the projective points do not lie on the ideal plane. As such, some choices of camera matrices may give poor results. The two methods produce identical projective structure estimates, so although none of our experiments have reproduced the aforementioned problem, we will stay with the original formulation.

In fact, our efforts to compare the two variations raised an interesting question: how to check the correctness of the resulting structure? A visual inspection of projective structure is of little use, a point made clear in figure 4.5. This shows the first pair of images from our quadrangle sequence, along with two views of the projective structure recovered by our system. It is quite obvious that the projective structure bears no resemblance whatsoever to the real scene. In general, this will be the case for any projective reconstruction.

In the absence of any other information there is no way to quantify the accuracy of the projective structure. The best we can hope for is to show that it is consistent, i.e. that it is actually a

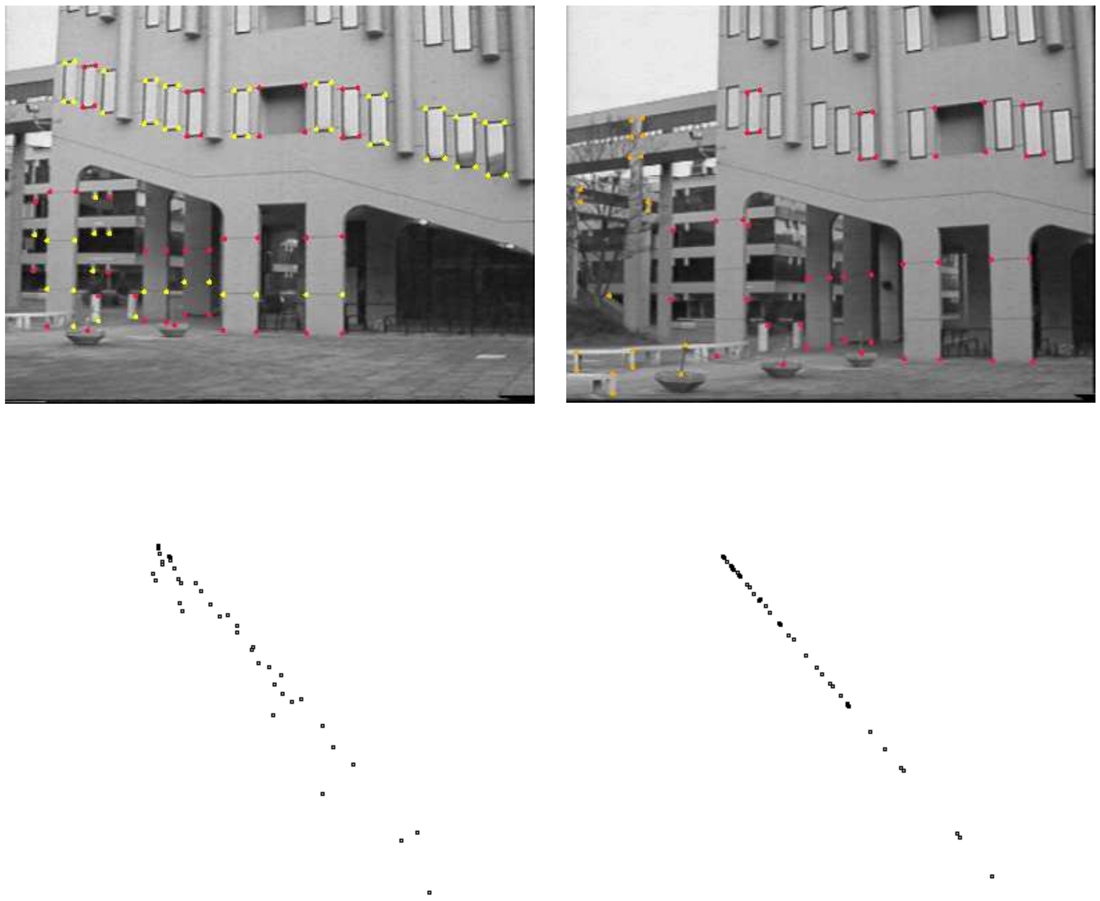


Figure 4.5: The first pair of images in the quadrangle sequence and two views of their projective reconstruction.

correct solution to the reconstruction problem. The simplest way to do this is to compute residuals for each system of linear equations. In other words, use the two camera matrices to *reproject* the 3D points and compare their reprojected image coordinates with those of the original matched features. We compute the ratio of the ratios of the original and reprojected x and y coordinates for each point pair. Ideally these values should all be 1, which provides an easy way to tell, at a glance, if the projective structure is correct. Also measured is the overall average distance, in pixels, between the original and reprojected image points, which one would hope to be close to zero.

Feature	Image 1					Image 2				
	Original		Reprojected			Original		Reprojected		
Label	x	y	x'	y'	Ratio	x	y	x'	y'	Ratio
0	159	236	159.0	235.5	0.998	246	256	246.0	256.0	1.000
8	218	167	218.0	167.4	1.002	307	184	307.0	184.0	1.000
16	32	135	32.0	134.8	0.998	108	155	108.0	155.0	1.000
24	100	226	100.0	226.1	1.000	174	247	174.0	247.0	1.000
34	77	139	77.0	139.0	1.000	132	159	132.0	159.0	1.000
60	255	60	255.0	59.9	0.999	343	71	343.0	71.0	1.000
76	142	61	142.1	62.1	1.018	222	77	221.0	77.0	1.000
85	70	78	70.0	78.3	1.004	146	96	145.0	96.0	1.000
95	46	5	46.0	5.1	1.025	117	22	116.0	22.0	1.000
109	17	74	17.0	73.7	0.997	90	93	90.0	93.0	1.000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Average Absolute Image Coordinate Differences										
			0.02	0.37				0.00	0.00	

Table 4.1: Results of reprojecting projective structure

Table 4.1 shows some example results of these computations for the image pair and projective structure of figure 4.5, using a representative sample of matched points having a range of coordinate values. The original, manually-detected, image points are shown to the nearest pixel, while the reprojected coordinates are rounded to the nearest tenth of a pixel. The reason for this sub-pixel accuracy is to illustrate that the original and reprojected coordinates are not necessarily identical. The ratio of ratios of the coordinates is given to three decimal places. Again, this is primarily to indicate that the ratio is not always exactly 1, but it also serves to show that the ratio of ratios is *furthest* from 1 when the x-coordinate is significantly larger than the y-coordinate, as with feature 95.

As can be seen, in this case all the coordinate ratios are equal, or very close, to 1, and the overall average coordinate differences are extremely small. Thus we can conclude that, regardless of what it *looks* like, the projective structure recovered by our system is indeed consistent with the given point matches and fundamental matrix. It is interesting to note the relative inaccuracy of the Image 1 reprojections. A possible explanation for this is that while the Image 2 camera was explicitly computed, the Image 1 camera matrix was fixed to $\mathbf{C} = (\mathbf{I}|\mathbf{0})$. Further investigation would be required to determine why the average Image 1 x and y-coordinates differences vary by more than a factor of ten.

4.7 Upgrading to Euclidean Structure

4.7.1 Method 1: A Direct Solution

The recovered projective structure is related to the real scene structure by a 3D projective transform. So, in order to obtain a full Euclidean reconstruction we need to determine the 4×4 transformation matrix, $\mathbf{H}$, which maps the set of projective points P_{Pi} to their Euclidean counterparts P_{Ei}. For one such pair of corresponding structure points, the mapping is

$$\begin{pmatrix} \alpha X_E \\ \alpha Y_E \\ \alpha Z_E \\ \alpha \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{pmatrix} \begin{pmatrix} X_P \\ Y_P \\ Z_P \\ T_P \end{pmatrix} \tag{4.17}$$

Multiplying out, cancelling the scale factor, α_i, and rearranging, gives the following set of three linear equations in the sixteen unknown elements of $\mathbf{H}$:

$$\begin{pmatrix} -X_P & -Y_P & -Z_P & -T_P & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X_EX_P & X_EY_P & X_EZ_P & X_ET_P \\ 0 & 0 & 0 & 0 & -X_P & -Y_P & -Z_P & -T_P & 0 & 0 & 0 & 0 & Y_EX_P & Y_EY_P & Y_EZ_P & Y_ET_P \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -X_P & -Y_P & -Z_P & -T_P & Z_EX_P & Z_EY_P & Z_EZ_P & Z_ET_P \end{pmatrix} \begin{pmatrix} H_{11} \\ \vdots \\ \vdots \\ H_{33} \end{pmatrix} = 0 \tag{4.18}$$

The projectivity $\mathbf{H}$ is only defined up to a scale factor (see Appendix A), and therefore has just fifteen degrees of freedom. Since each correspondence $P_{Pi} \Leftrightarrow P_{Ei}$ gives rise to three such equations, we only need five ground truth points to solve for the elements of $\mathbf{H}$. This transformation can then be applied to *all* the projective points to obtain Euclidean structure.

4.7.2 Method 2: An Indirect Solution

An alternative method for obtaining the projectivity has been suggested which does not rely directly on the projective structure [23]. Just as $\mathbf{H}$ maps from projective to Euclidean structure, its inverse can be used to perform the reverse transformation i.e. $\mathbf{P_P}_i = \mathbf{H}^{-1}\mathbf{P_E}_i$. Thus we can formulate a set of constraints on the elements of $\mathbf{H}^{-1}$, based on the perspective projection equations

$$\begin{pmatrix} \alpha_i u_i \\ \alpha_i v_i \\ \alpha_i \end{pmatrix} = \mathbf{CH}^{-1} \begin{pmatrix} X_{Ei} \\ Y_{Ei} \\ Z_{Ei} \\ T_{Ei} \end{pmatrix} \tag{4.19}$$

and

$$\begin{pmatrix} \beta_i u'_i \\ \beta_i v'_i \\ \beta_i \end{pmatrix} = \mathbf{C}'\mathbf{H}^{-1} \begin{pmatrix} X_{Ei} \\ Y_{Ei} \\ Z_{Ei} \\ T_{Ei} \end{pmatrix} \tag{4.20}$$

Multiplying out, cancelling the scale factors, α_i and β_i, then rearranging, gives a set of four, rather involved, linear equations for each combination of point matches and corresponding ground truth. However, only three of these are linearly independent, so we need at least five such combinations to solve for $\mathbf{H}$, up to a scale factor. We can then invert, to obtain $\mathbf{H}$ and upgrade our projective structure as before.

4.7.3 Initial Experiments

In order to compare the performance of the two methods, described above, we carried out a series of simple experiments. Although only five ground truth points are actually needed to solve for $\mathbf{H}$, it helps to have more available, to be able to measure the accuracy of the projectivity with respect to how it transforms points which were not used in its computation. For these experiments a set of twelve[3] ground truth points P_{Ei} was used, corresponding to matched points in the quadrangle images, and chosen to give a good distribution in the 3D world space.

First of all, a random subset of five of the ground truth points was selected. This was used to estimate projective transform matrices, $\mathbf{H}_1$ and $\mathbf{H}_2$, for each of the two methods. Next, the projectivities were applied to *all*[4] of the projective points, to obtain new estimates for their Euclidean locations $\mathbf{H}_1\mathbf{P}_{Pi} = \hat{\mathbf{P}}_{E1_i}$ and $\mathbf{H}_2\mathbf{P}_{Pi} = \hat{\mathbf{P}}_{E2_i}$. Finally, measures of the quality of the projectivities, Q_{H_1} and Q_{H_2}, were computed as the average Euclidean distance between the twelve estimated Euclidean points and the corresponding ground truth $dist(\hat{\mathbf{P}}_{E1_i}, \mathbf{P}_{E_i})$ and $dist(\hat{\mathbf{P}}_{E2_i}, \mathbf{P}_{E_i})$.

The experiment described above was performed 10000 times and the Q_H values for the two methods were examined. There was little correlation between the two sets of values. That is, a subset of correspondences that generated an accurate projective transform using the direct method would not necessarily do so for the indirect method. We have not performed a full statistical analysis of these results and so can only state that, on a trial for trial basis, the direct method outperformed the indirect method on approximately 68% of occasions.

The sets of Q_H values were sorted and Figure 4.6 shows a graph of the 100 *lowest* values obtained for each method. The ground truth points for this pair of images were spread over a world space volume of approximately $50\times10\times50$ metres, so both methods exhibit good best-case

[3]This is in fact just a subset of over a hundred ground truth points for the initial images of the quadrangle sequence. These were obtained by the author, with the aid of a trusty tape-measure, one cold, dark, miserable Sunday evening, when the campus was, thankfully, deserted!

[4]In these experiments a total of 43 projective/Euclidean correspondences were available.

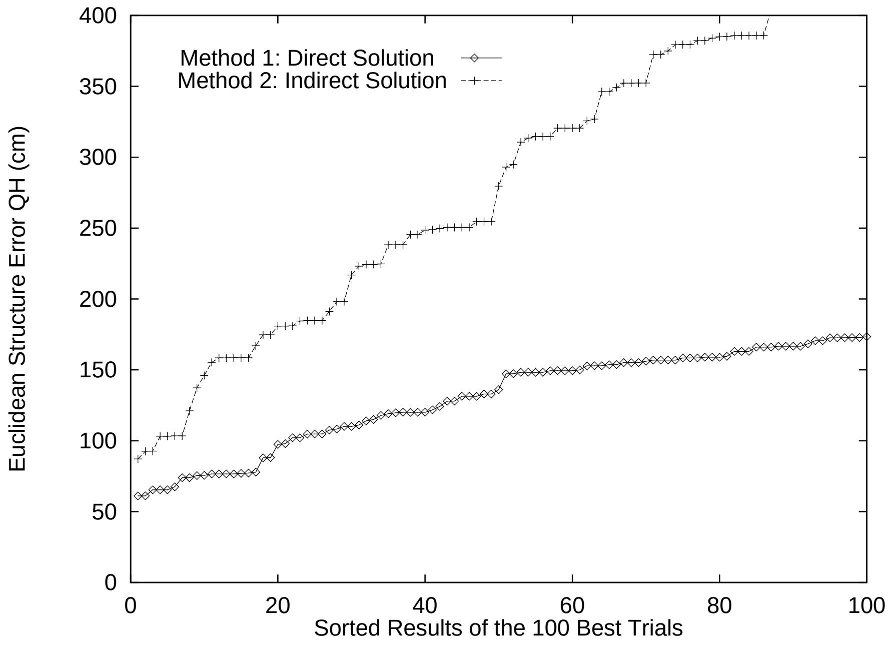


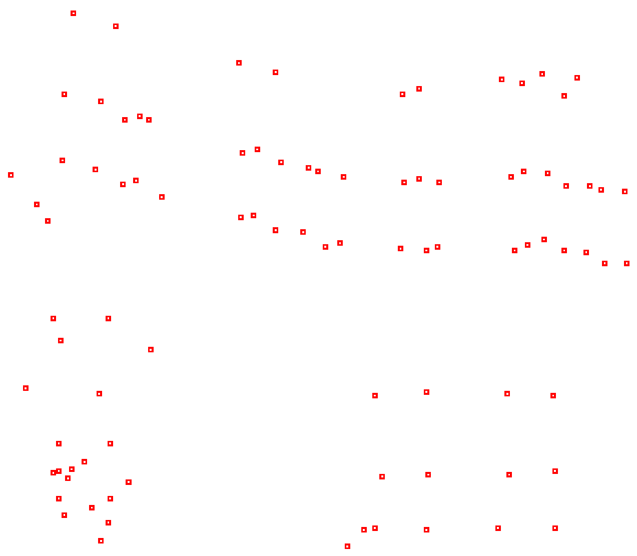
Figure 4.6: Comparison of the accuracy of the direct and indirect methods for computing the transform from projective to Euclidean structure.

performances, getting the average Euclidean distance error down to well under a metre. However, it is clear that, on average, the direct method gives the most accurate results. In the sequel, all our discussions and experiments will be based on this method of solving for the projective transform.

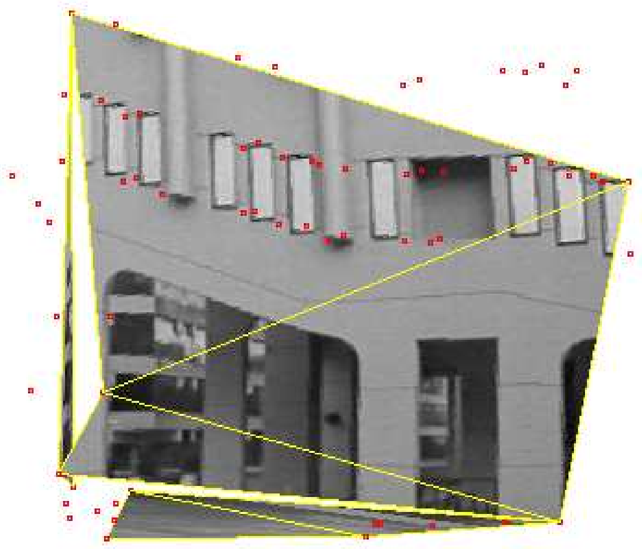
Figures 4.7–4.10 show the Euclidean structure recovered by this method, using the trial above which had the lowest Q_H value, for the lecture theatre building in the image pair of figure 4.5. As it is rather difficult to portray three-dimensional structure on a two-dimensional page, especially using just point features, several different views of the reconstruction are shown. A partial triangulation of some of the points into 3D facets is performed, and subsequent texture-mapping[5] makes it easier to see the relationship between the real and recovered structure. These results demonstrate that it is possible to obtain a good estimate for the Euclidean structure of a projective scene, if just five ground truth points are known.

[5]The texture-mapping process is very basic, simply interpolating the pixel values from the original image between the triangle vertices. Since camera calibration is unknown, there is no attempt to correct for camera motion, and thus the texture-mapping only looks realistic when the structure is viewed from close to the original camera position.

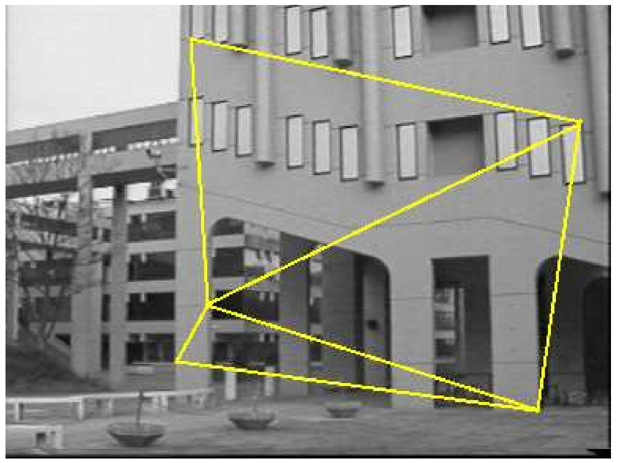
Figure 4.7: (a) Front view of the reconstructed points



(b) Partial triangulation with texture-mapping

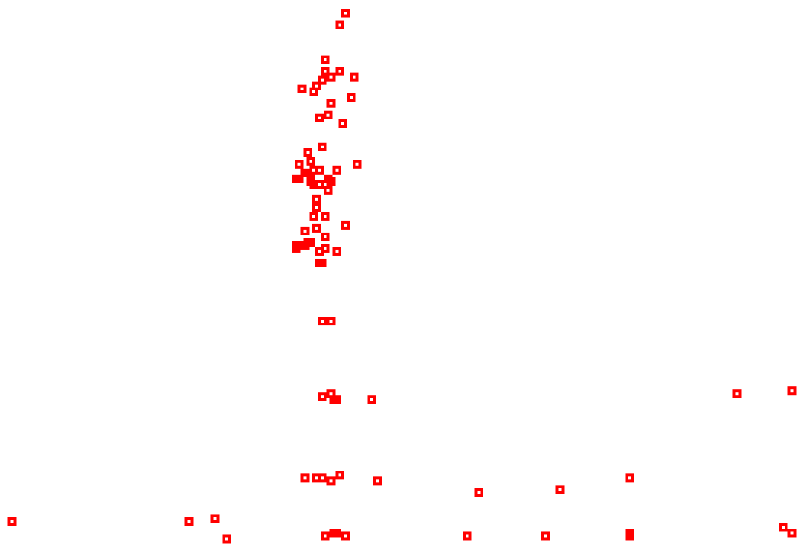


(c) Texture triangles in the original image



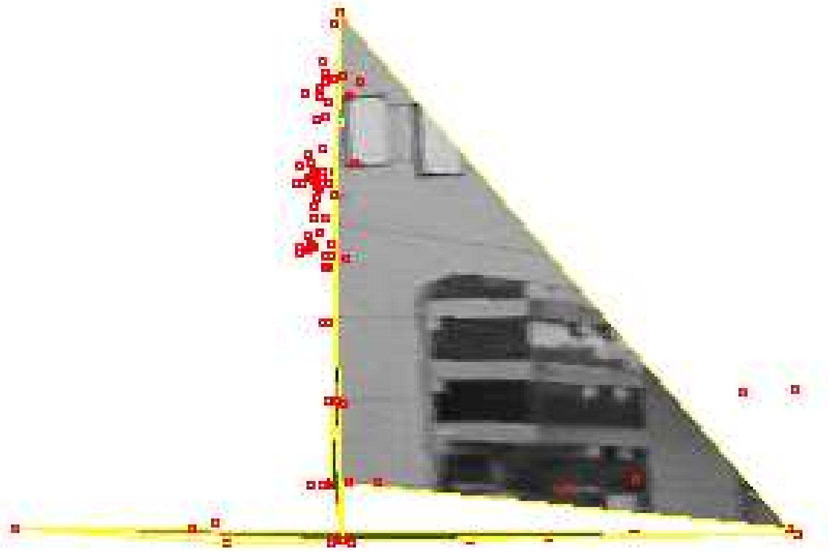
This view of the reconstruction is from a position close to the original camera location, looking directly towards the lecture theatre building. The 'staircase' of points in the top half of (a) corresponds to the corners of windows on the front face of the building, while the 12 points at the bottom right belong to the middle and right supporting pillars. In (b), the front face of the building has been modelled using 3 texture-mapped triangles. The original image locations of these triangles are shown in (c). The thin triangles at the extreme left and bottom of (b) correspond to the left side of the building and the ground plane, respectively. These are easier to see in figures 4.8 and 4.9, which show the reconstruction from different viewpoints.

Figure 4.8: (a) Side view of the reconstructed points



This view of the recovered structure is from a position at the right-hand side of the lecture theatre building, looking directly along the plane of its front wall. The features on this wall (corners of windows etc.) form a vertical band of points in (a), while features on or near the ground plane form a perpendicular group. The triangular facet in (b) is part of the left-hand wall of the building, but the texture-mapping is not consistent with the viewpoint due to the significant change in camera position. The original image location of the texture triangle is shown in (c).

(b) Partial triangulation with texture-mapping



(c) Texture triangles in the original image

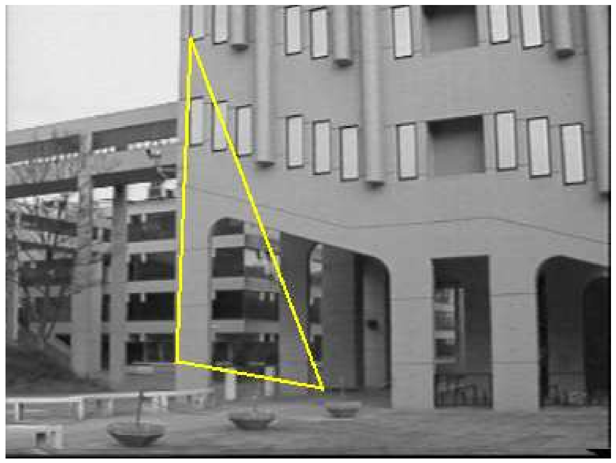
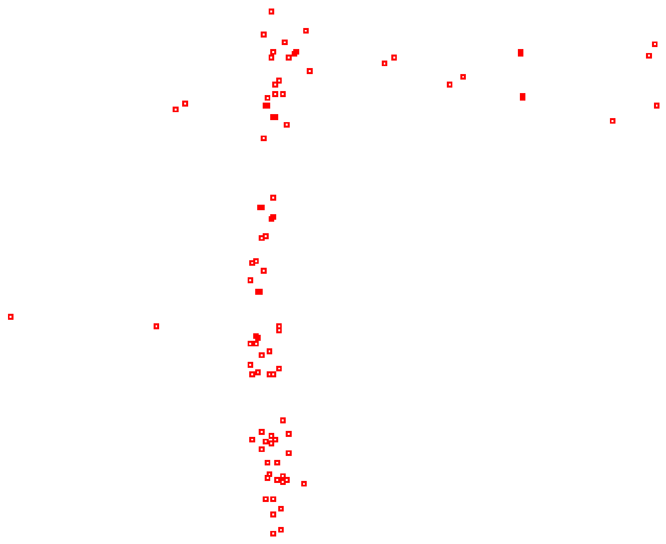
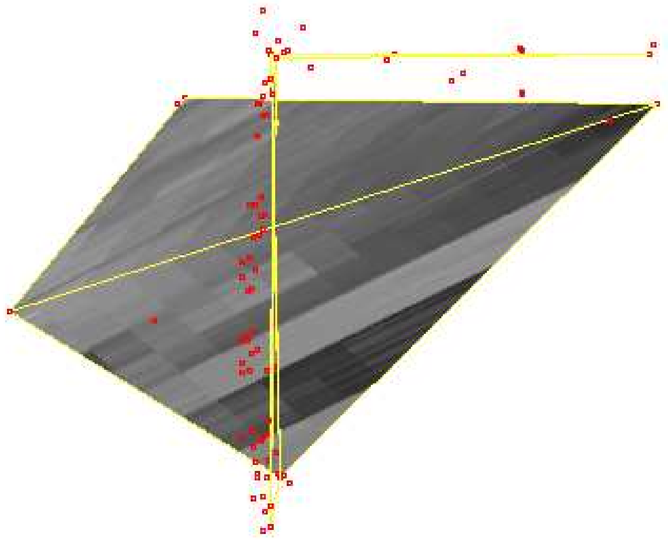


Figure 4.9: (a) Top view of the reconstructed points



(b) Partial triangulation with texture-mapping



(c) Texture triangles in the original image

This is a top-down view of the recovered structure of the lecture theatre. The vertical band of points in (a) represents features on the front wall of the building, while the group of points at the top-right belongs to the pillars and bollards at the left side of the building. Clearly visible in (b) is the right-angled structure formed by the triangulations of some of these points (as shown in figures 4.7 and 4.8). The kite-shaped polygon, consisting of two triangles, depicts the ground plane. The original image locations of the texture triangles are shown in (c).

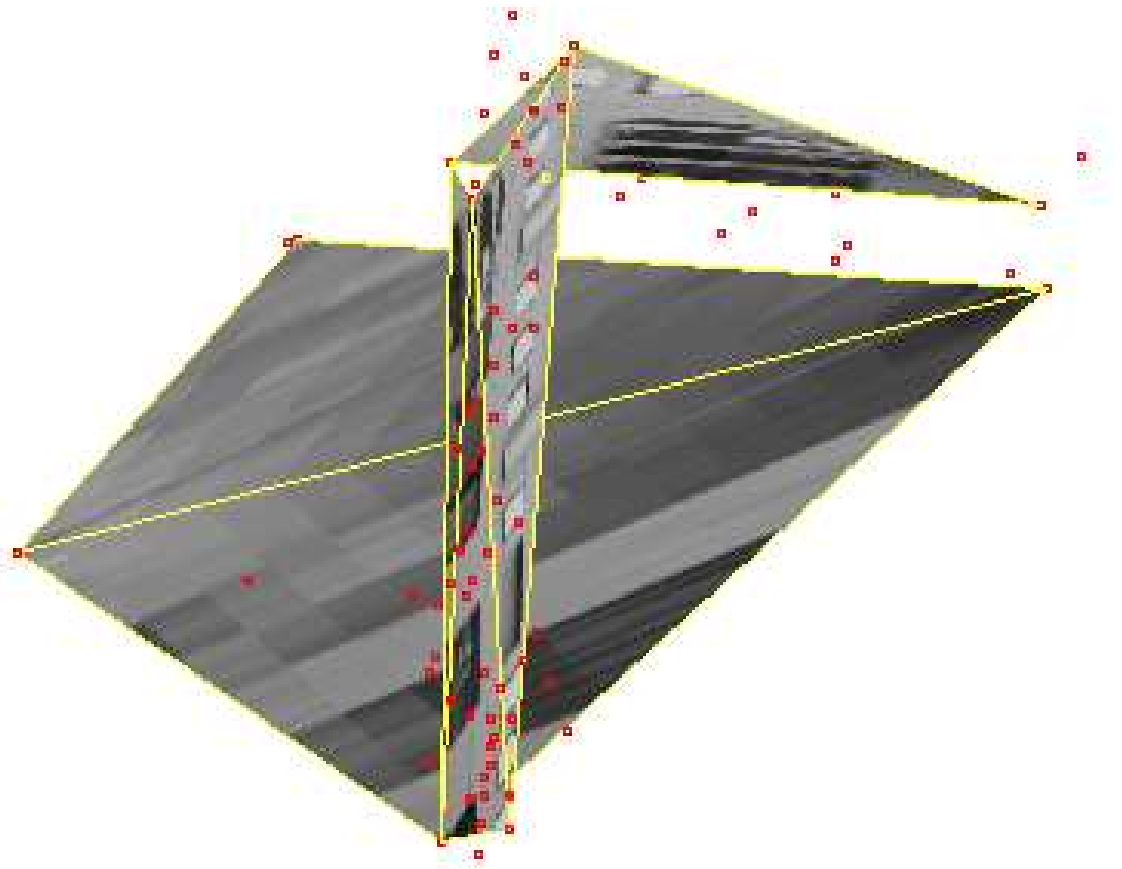


Figure 4.10: This is another top-down view of the lecture theatre reconstruction. The viewpoint is slightly different to that in figure 4.9 to show more of the texture-mapping on the sides of the building and give a better sense of the recovered structure.

4.8 Incremental Reconstruction

Once a Euclidean structure estimate has been acquired for the scene in the first image pair, it can be extended incrementally by processing each remaining image in the sequence. The initial steps of this processing are identical to the ones used for the first image pair i.e.

1. Load the next image in the sequence and call it the current image.
2. Obtain a set of matched features in the current and previous images.
3. Use the matched features to estimate a fundamental matrix.
4. Factorise the fundamental matrix to obtain a pair of camera matrices.
5. Use the camera matrices to determine a *projective* reconstruction for the *matched* points.

At this point we calculate the projective transform to upgrade to Euclidean structure, as before. However, there is a difference: *we no longer rely upon the user to supply the system with ground truth data*. If the newly-computed projective structure includes at least five points for which there exist previously estimated Euclidean coordinates, then these correspondences are used to obtain a projective transform. This is then be applied to the remaining[6] projective structure to map it into the existing Euclidean frame. The whole process is repeated for each subsequent image in the sequence, at each step 'stitching' new and old structure together via the projectivities. Thus a Euclidean model of the entire scene is gradually built, from a basis of as few as five ground truth points.

This algorithm places some restrictions on the visibility of features between images in the sequence. Our incremental Euclidean reconstruction process requires five or more projective points which have already been assigned Euclidean 3D coordinates. In order for this to be true, these points must previously have taken part in the process of fundamental matrix calculation, projective reconstruction and upgrade to Euclidean structure. Hence, at some point they must have been included in a set of eight or more matches between two successive images. Thus, each extension

[6]Those points without Euclidean correspondences.

to our overall Euclidean structure requires five points which have been viewed in at least three images. These constraints have been built into our GUI-based feature detection/matching system, so at each step the user knows exactly how many image matches or 3D correspondences are required to proceed.

There is one other implementation detail we have not yet discussed. So far, we have been happy to assume that the projectivity calculation could be performed simply by choosing the best result from a small number of trials. This approach is perfectly valid for the first pair of images in the sequence, when we know the correspondences between projective points and accurate ground truth data. This accuracy also means that there is little to be gained from using more than five of the ground truth points. However, when processing the remainder of the sequence, the projectivity is computed using a set of previously *estimated* Euclidean points. Obviously, some of these will be more accurate than others and, hopefully, there will be many more than five correspondences. We would like to devise an efficient way to choose the best subset of these with which to calculate the projective transform. The is reminiscent of the problem faced in section 3.7.1, when attempting to estimate the fundamental matrix from a noisy set of matched image points. Here, we employ a similar solution.

4.8.1 A RANSAC Approach

We have embedded the direct method of computing the projective transformation matrix, $\mathbf{H}$, into a **RANSAC*** algorithm. Given a set of five or more correspondences between projective and existing Euclidean structure, we perform a series of trials. In each trial, the first step is to select, at random, a 5-element subset, from all the correspondences. This is used to compute the projectivity which is then applied to *all* the projective points to obtain new estimates for their Euclidean positions. We then compute the distance between each newly estimated and existing Euclidean points. Those correspondences for which this distance is below a given threshold form the *consensus set*. The (≥ 5) correspondences in the consensus set are then used to calculate another projectivity, for which we compute the average Euclidean distance error, over all correspondences, as in section 4.7.3. The old consensus set is replaced if the new one is of equal or larger size and has a lower Q_H value. We then move on to the next trial and the process continues until either all the trials have been completed or the consensus set size or Q_H value reach a specified target. In detail, the **RANSAC*** projectivity calculation is as follows:

Given five or more projective⇔Euclidean structure correspondences $\{P_{P_i}, P_{E_i}\}$, then for some fixed number of trials:

1. Randomly select a subset of 5 correspondences, $P_{5C} \subset \{P_{P_i}, \hat{P}_{E_i}\}$.
2. Use P_{5C} to compute the projectivity $\mathbf{H}$.
3. Apply $\mathbf{H}$ to obtain new Euclidean structure estimates $\mathbf{H}\mathbf{P}_{P_i} = \hat{\mathbf{P}}_{E_i}$.
4. Determine the consensus set C of correspondences whose Euclidean distance error is below some threshold.
5. Calculate a new projectivity, $\mathbf{H}'$, based on C.
6. Compute Q_H for $\mathbf{H}'$.
7. Replace the old consensus set if C is the same size or larger, and has a lower Q_H.
8. Terminate if the size of C or Q_H reach specified targets.

At the end of this procedure, we hope to have obtained a projective transform, based on many of the correspondences, with a low Q_H value, and to have done so in a relatively small number of trials.

4.8.2 Results

Figure 4.11 shows the results of the reconstruction algorithm after processing the initial ten frames of the quadrangle sequence. Although the two-dimensional constraints of the page make it difficult to get a feel for the recovered three-dimensional structure, this plan view indicates how the structure has been extended from that shown in figure 4.9. Clearly visible at the middle right of the diagram, is the right-angled corner of the building, which is in view over the first few frames. The set of points leading away from this, to the south, corresponds to a series of bollards, while the long straight, east-west configuration represents the struts of a continuous concrete bench. The set of points at the very top of the figure consists of features on the vertical surface of a distant building. The remaining points in the centre of the plan belong to various prominent scene features. Although these do not provide much in terms of readily identifiable scene structure, they are a valuable source of extra data for satisfying the constraints of section 4.8.

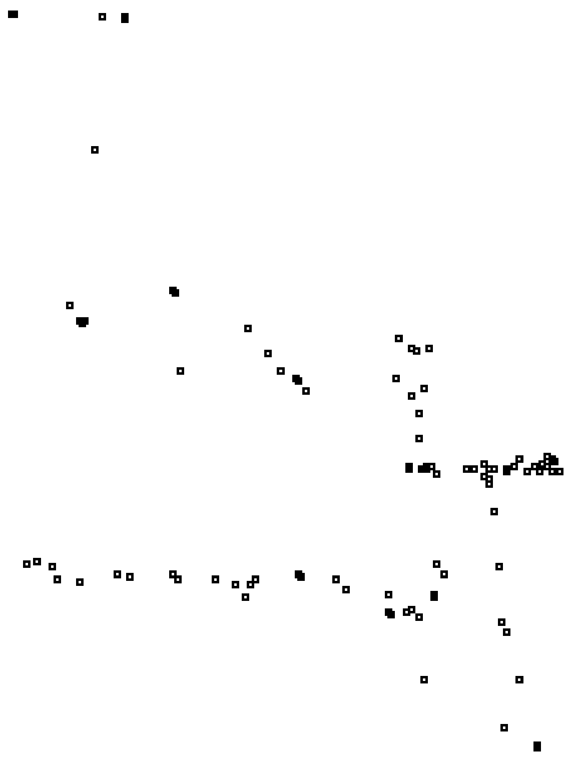


Figure 4.11: Plan view of the recovered quadrangle scene after 10 image pairs.

Although the algorithm performs well at recovering the structure of objects which are close to the camera, objects at long range cause problems. The reason for this is the image resolution, which is too low to allow feature points on distant objects to be detected and matched with sufficient accuracy. Take, for example, the point at the top left of figure 4.11, which is approximately fifty metres from the camera. Changing the coordinates of one of its corresponding image points by just a single pixel resulted in a *seven metre* shift in its recovered Euclidean structure. These erroneous structure points can have a detrimental effect on the overall reconstruction, if they are used in the calculation of the projective transform at later steps. For example, figure 4.12 shows a plan view of another reconstruction of the same section of the quadrangle scene, which has failed for this reason; the points at the upper left of the figure being completely wrong. At the moment, this problem is avoided by only considering foreground structure when performing manual feature detection and matching.

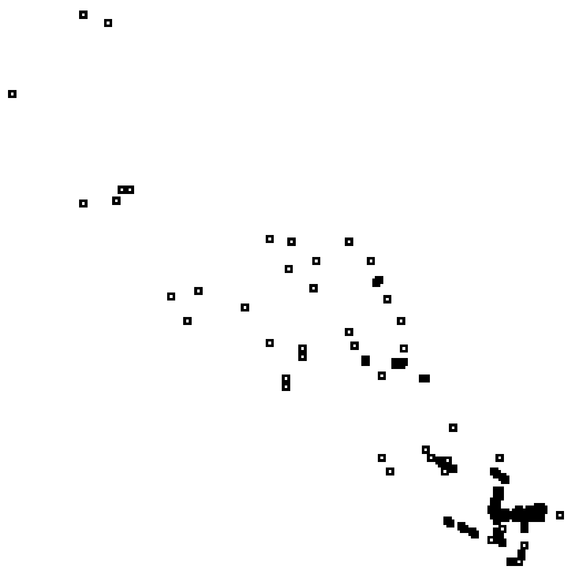


Figure 4.12: Failed reconstruction of part of the quadrangle scene

Results indicate that at each step in the reconstruction process, the error in the recovered Euclidean structure increases, even that of objects which are close to the camera. This is to be expected, due to the knock-on effect of calculating the projective transform at each step using previously *estimated* Euclidean structure. In the case of a long image sequence, the accumulated error could become very large, leading to inconsistencies in the recovered structure. In the next chapter we will describe our attempts to deal with this problem.

Chapter 5

Structure Recognition and Matching

5.1 Introduction

Our experiments of the previous chapter have shown that errors in the recovered Euclidean structure can accumulate rapidly, over the course of a few frames, and even when the reconstruction process is initialised with ground truth data. We must consider the effect this would have on a long sequence of images, especially one in which the camera eventually returns to view a part of the scene that has been seen previously. For example, figure 5.1 shows a simple diagrammatic representation of such an occurrence: a complete walk-around of a rectangular building.

In this example, the reconstruction process starts by recovering structure for the front of the building. With our reconstruction system, this structure is extended incrementally as the viewer walks clockwise around the building, until eventually, the camera returns to somewhere close to its starting position. At this point it will be viewing the front of the building for the second time and thus we will have obtained two different sets of Euclidean structure for this section of the building. The errors in the structure recovery process will mean that the two structure segments are not aligned correctly in the world coordinate frame. If our reconstruction system is to be of any practical use, we need to find a way to turn this initial incorrect structure into one which is internally consistent.

In order to do this, two problems must be solved. First of all, we must be able to recognise

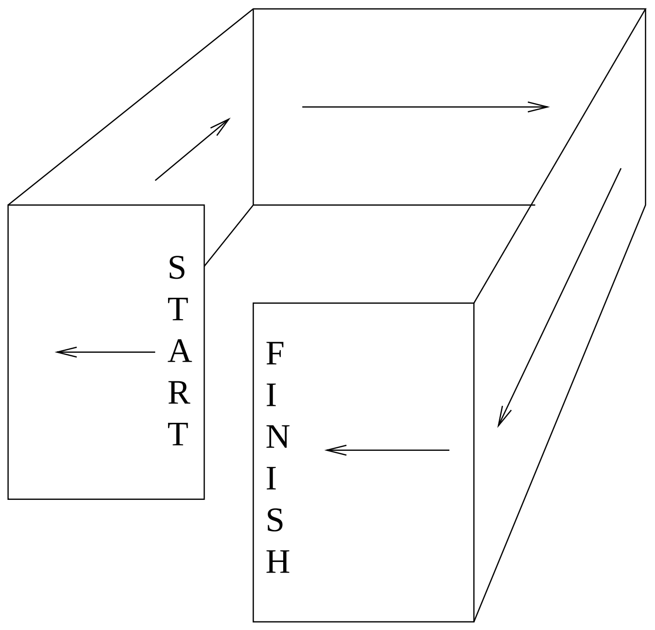


Figure 5.1: Errors in the reconstruction process can lead to internally inconsistent structure.

when two recovered structure segments actually correspond to the same part of the real-world scene. Once this has been accomplished, it is necessary to find the transformation which brings the two structure segments into alignment. In terms of model-based recognition, these are the problems of indexing and matching. The task is further complicated by the fact that occlusion and/or a change of viewpoint could mean that the two segments do not contain exactly the same points. We show that these problems can be solved using the *geometric hashing* paradigm, a general method for model-based object recognition. It was originally developed for the task of identifying flat rigid objects [35, 34] from images, using an affine approximation to the full perspective camera, but the same approach can be used for many recognition problems involving a variety of transformations in two and three dimensions [37].

Classical geometric hashing is a two-stage process, *object representation* and *matching*. Briefly, the first stage, which is usually done off-line, involves the construction of a hash table, to provide an invariant representation for each of a series of model objects that are to be recognised. In the second stage, features extracted from images of the scene are used to compute invariants, with which to index the hash table and tally votes for candidate models i.e. those that could feasibly be present in the scene. A high vote count not only indicates the likely presence of an instance of a particular model object in a view of the scene, but the corresponding invariants determine the model$\rightarrow$view transformation.

The same basic algorithm can be used to solve our problem of matching two segments of Euclidean structure, with one important difference. Rather than having a priori knowledge of a set of model objects to be recognised, our hash table starts off completely empty. As the incremental reconstruction system recovers new estimated Euclidean points, these are hashed into the table. In a sense, the system 'learns' its own models of segments of scene structure as it goes along. Later in the image sequence, if we come across the same structure segment again, the geometric hashing system should recognise this, *and* provide the transform between the old and new segments, thus solving both our indexing and matching problems. As with all indexing techniques, geometric hashing allows for the recognition of many models simultaneously. This is important for our application, in which an extended environment could be represented by hundreds of structure segment models.

We begin this chapter with an overview of the original geometric hashing paradigm, followed by a discussion of the details of our implementation. We describe our method for representing 3D structure in a form which is invariant to Euclidean transformations, and present a simple, symmetry-based approach for improving the efficiency of the hashing process. Finally, we describe the way we have embedded the geometric hashing algorithm into our structure recovery system and discuss the initial results of experiments on real image sequences.

5.2 An Overview of Geometric Hashing

In this section we review the geometric hashing paradigm in the context of its original application: the recognition of flat objects under the affine transformation (rotation, translation, scale and shear). This allows for a very simple explanation of the algorithm's main concepts and will help us to show, later in the chapter, how our implementation differs from that of the standard method. The general scheme of the geometric hashing paradigm is shown in figure 5.2.

When viewing flat objects which are relatively distant from the camera, orthographic projection (with a scale factor), provides a good approximation to the full perspective camera [29].

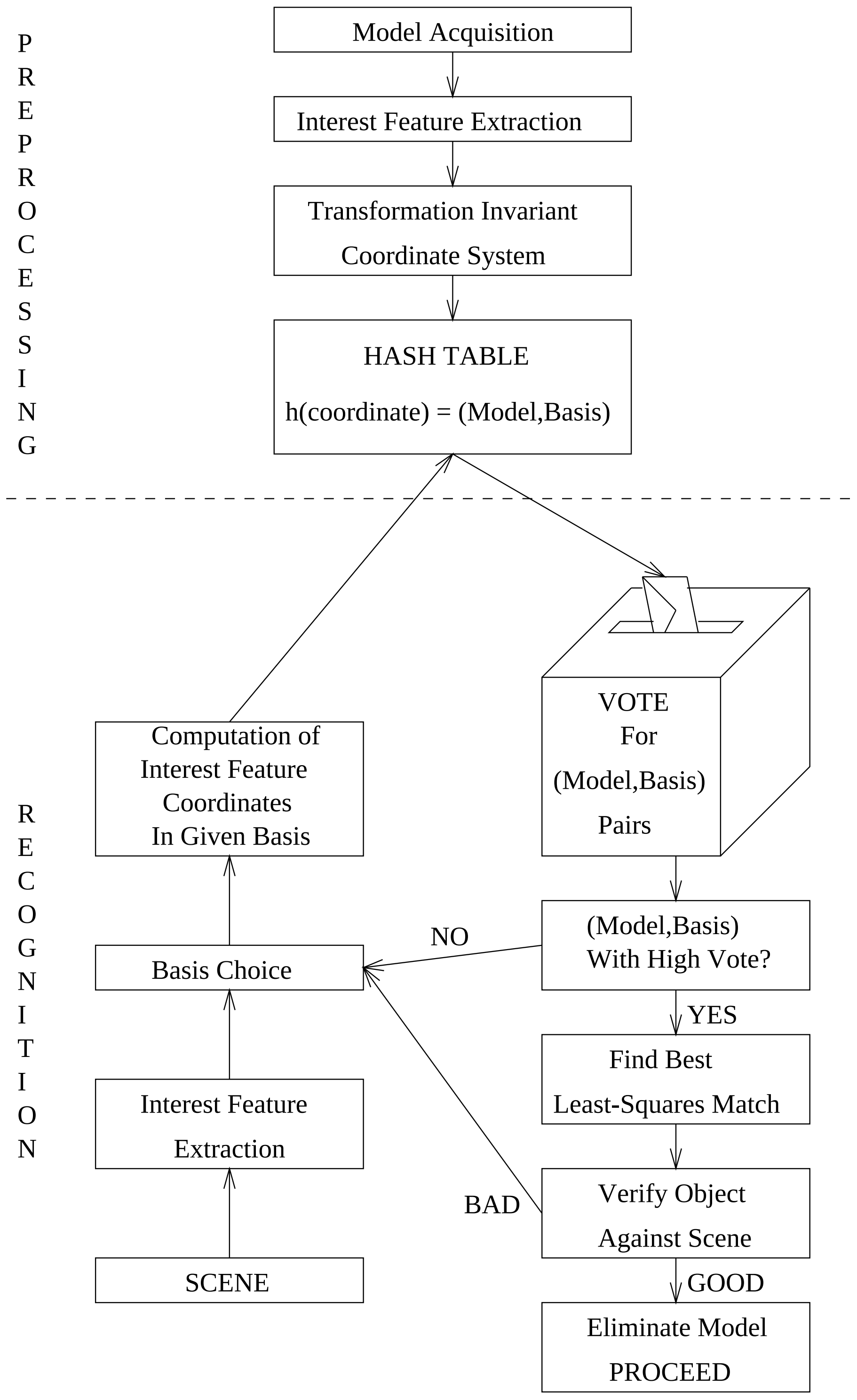

PREPROCESSING
Model Acquisition
Interest Feature Extraction
Transformation Invariant Coordinate System
HASH TABLE
h(coordinate) = (Model,Basis)
RECOGNITION
VOTE For (Model,Basis) Pairs
Computation of Interest Feature Coordinates In Given Basis
(Model,Basis) With High Vote?
NO
YES
Basis Choice
Find Best Least-Squares Match
Interest Feature Extraction
BAD
Verify Object Against Scene
SCENE
GOOD
Eliminate Model PROCEED

Figure 5.2: The general scheme of the geometric hashing algorithm

Under this assumption, two different images of the same flat object are related by a 2D affine transformation $\mathbf{T}$. In other words, there exist a non-singular 2×2 matrix $\mathbf{A}$ and 2D vector $\mathbf{b}$, such that a non-homogeneous point p in the first image is transformed to the corresponding point $\mathbf{Ap}+\mathbf{b}$ in the second. The transformation has six degrees of freedom and can thus be fully determined by three point correspondences. Assuming that sets of interest points have been extracted from images of the model objects and the scene, the problem becomes that of determining if some transformed subset of scene points matches a subset of any of the model point-sets. The key to this lies in representing the point sets in a manner which is invariant to the affine transform.

5.2.1 2D Affine Invariance

Suppose we have extracted a set of m points from an image of one of our model objects. We can pick any ordered, non-collinear triplet of points from this set and use them to represent all the other points. Consider figure 5.3. Let a_{00}, a_{10} and a_{01} be three, non-collinear points in the image. The vectors $\mathbf{i} = (\mathbf{a}_{10} - \mathbf{a}_{00})$ and $\mathbf{j} = (\mathbf{a}_{01} - \mathbf{a}_{00})$ are linearly independent, hence they are a 2D linear *basis*. Any point p in the image can be represented as a linear combination of these two basis vectors. In other words, there is a pair of scalars (α,β) such that:

$$\mathbf{p} = \alpha\mathbf{i} + \beta\mathbf{j} + \mathbf{a}_{00} = \alpha(\mathbf{a}_{10} - \mathbf{a}_{00}) + \beta(\mathbf{a}_{01} - \mathbf{a}_{00}) + \mathbf{a}_{00} \tag{5.1}$$

Application of the linear affine transform $\mathbf{T}$ to this representation for p gives

$$\mathbf{Tp} = \alpha(\mathbf{Ta}_{10} - \mathbf{Ta}_{00}) + \beta(\mathbf{Ta}_{01} - \mathbf{Ta}_{00}) + \mathbf{Ta}_{00} \tag{5.2}$$

Hence the transformed point $\mathbf{Tp}$ has the same coordinates (α,β) with respect to the affine basis $\mathbf{Ta}_{00}\mathbf{Ta}_{10}\mathbf{Ta}_{01}$ as has $\mathbf{p}$ with respect to $\mathbf{a}_{00}\mathbf{a}_{10}\mathbf{a}_{01}$. For example. suppose we make the following coordinate assignments for the points in figure 5.3: $\mathbf{a}_{00} = (5, 1)$, $\mathbf{a}_{10} = (10, 1)$, $\mathbf{a}_{01} = (7, 5)$ and $\mathbf{p} = (24, 9)$. Writing these in terms of equation 5.1 gives:

$$\begin{pmatrix} 24 \\ 9 \end{pmatrix} = \alpha \underbrace{\begin{pmatrix} 5 \\ 0 \end{pmatrix}}_{\mathbf{i}} + \beta \underbrace{\begin{pmatrix} 2 \\ 4 \end{pmatrix}}_{\mathbf{j}} + \begin{pmatrix} 5 \\ 1 \end{pmatrix} \tag{5.3}$$

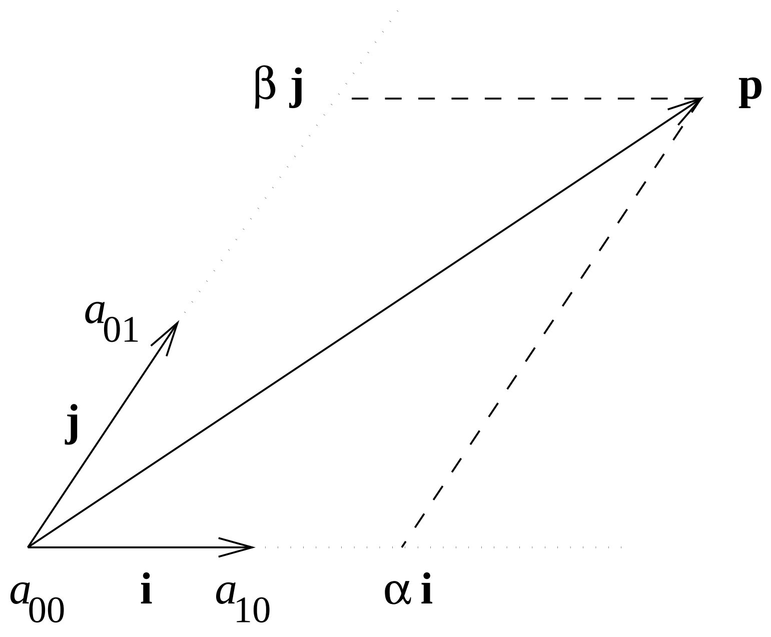


Figure 5.3: Representing p using the affine basis triplet $a_{00}a_{01}a_{10}$.

which means, by inspection $\alpha = 3$ and $\beta = 2$. Now, suppose we have an affine transform $\mathbf{T} = \mathbf{Ax} + \mathbf{b}$ where

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \qquad \text{and} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{5.4}$$

Applying this to our four points gives: $\mathbf{Ta}_{00} = (10, 15)$, $\mathbf{Ta}_{10} = (15, 25)$, $\mathbf{Ta}_{01} = (20, 23)$ and $\mathbf{Tp} = (45, 61)$. Once again, in terms of equation 5.1 we have:

$$\begin{pmatrix} 45 \\ 61 \end{pmatrix} = \alpha \underbrace{\begin{pmatrix} 5 \\ 10 \end{pmatrix}}_{\mathbf{i}} + \beta \underbrace{\begin{pmatrix} 10 \\ 8 \end{pmatrix}}_{\mathbf{j}} + \begin{pmatrix} 10 \\ 15 \end{pmatrix} \tag{5.5}$$

Clearly $\alpha = 3$ and $\beta = 2$ is a solution to this simple pair of linear equations. Thus we have obtained a representation for the point p which is invariant to the affine transform $\mathbf{T}$. Armed with the mechanism for computing such invariants, we proceed with a description of the two steps that make up the geometric hashing method: model representation and matching.

5.2.2 Model Representation

Classical geometric hashing usually begins with a preprocessing step, to generate a hash table containing invariant descriptions of the model objects that the system needs to be able to recognise. In the 2D affine case, this implies we have a database of images of our model objects, and that from each of these we have extracted sets of *interest points* corresponding to model features. The model representation is constructed by considering every possible three point subset of interest points as an affine basis and, in each case, hashing the invariants computed for all the remaining points. The outline for this preprocessing stage is as follows:

For each model:

1. Extract a set of m interest points.
2. For each ordered, non-collinear triplet of interest points (affine basis):
   - Compute the invariant coordinates of the remaining $m-3$ interest points in the affine frame defined by the basis triplet.
   - Pass each set of invariant coordinates to a *hash function* which generates indices into the hash table.
   - At each given hash table location, known as a *bucket*, store a record of the model and affine basis from which the invariants were obtained. Note that the finite size of the hash table will often lead to *collisions*, whereby more than one (model,basis) pair needs to be stored in each bucket. In such cases the bucket holds a list of these pairs (see figure 5.4).

The preparation of the hash table can be looked upon as a kind of learning process, in which various different representations of the models are memorised. Since it requires no knowledge about the scenes in which the models are to be recognised, it is usually performed off-line. Once preprocessing is completed, the geometric hashing system is ready for the matching stage.

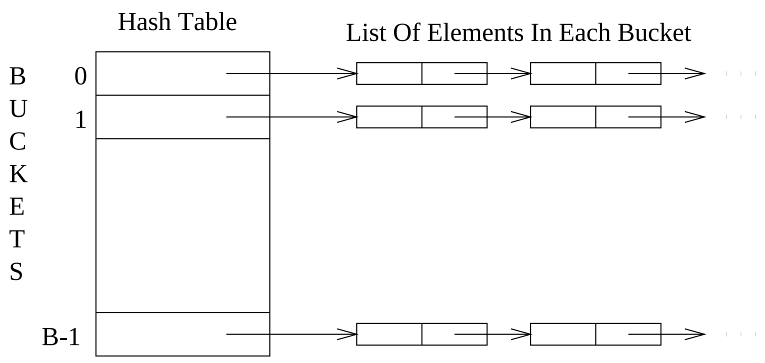


Figure 5.4: Hash table organisation

5.2.3 Matching

In geometric hashing the object representation and matching stages follow along very similar lines. When matching, we are presented with an image of a scene, and wish to determine which, if any, of our model objects are currently in view. To do this we:

1. Extract a set of n interest points from the image.
2. Choose an arbitrary ordered triplet of non-collinear points and use them as a basis with which to compute the affine invariant coordinates of the remaining $n - 3$ points.
3. Pass each set of invariant coordinates to the hash function which generates indices into the hash table.
4. Check each indexed hash table bucket and tally a vote for every (model,basis) pair stored there.
5. Look for a (model,basis) pair which scores a high number of votes. Each such pair implies that an instance of the given model is present in the scene. Furthermore the uniquely defined affine transformation between the *candidate* model and image basis triplets is assumed to be the transformation that maps between the model and the scene.
6. Apply the transformation obtained in step 5 to all the image points that voted for the candidate model to induce additional model-image point correspondences. Find the best transformation between all the correspondences, in a least-squares sense.

7. Transform the entire low-level representation of the model (which may include additional information, such as edge features, colours etc.) via the affine transformation obtained in step 6 and verify it against the scene. If the verification confirms the existence and orientation of the model, the matching process is complete. If this candidate solution is rejected, another one from step 5 is examined. If there are no more candidate solutions, go back to step 2.

It should be noted that, in general, the voting scheme will not result in just one candidate solution. In fact, that is not really the aim. Rather, the intention is to reduce significantly the number of candidates which make it through to the verification step. Also, since votes are cast for all models simultaneously, the complexity of the recognition process is independent of the size of the model database.

One of the nice features of geometric hashing is its ability to recognise partially occluded objects. This is made possible by the preprocessing stage, which constructs model representations using all the basis triplets from the points of interest. Thus, for matching to succeed it is enough to pick three points in the scene which belong to some model, in which case the appropriate (model,basis) pair will score highly in the voting procedure.

5.3 Hashing Euclidean Structure

The description of geometric hashing in the previous sections was based around the problem of recognising models and scenes related by a 2D affine transform. As we have seen, computing invariants under such a transform requires a three-point basis. Other transformations have different basis requirements. For example, recognition of objects which have undergone a 2D or 3D translation needs only a one-point basis, and a four-point basis for a projectivity between two planes [37]. A projectivity from $\mathcal{P}^3$ to $\mathcal{P}^3$ requires a five-point basis [12]. In our case, we wish to match two sets of 3D Euclidean structure, which differ by some rigid transformation. It will be shown that this problem can be solved using a three-point[1] basis.

[1]In fact, this is also the case for the 3D similarity transform, so we can solve for this more general problem with no additional complexity.

In passing, we note that given a set of n model/scene interest points and a k-point basis, the worst case complexity of the geometric hashing algorithm is $O(n^{k+1})$ [37]. Thus, by upgrading from projective to Euclidean structure as part of the reconstruction process, the complexity of the recognition task has been reduced from $O(n^6)$ to $O(n^4)$.

5.3.1 Computing Euclidean Invariants

Suppose we have a set of four 3D Euclidean structure points. Three of these can be used to define a new coordinate system (basis) in which the position of the fourth point is invariant to a 3D rigid or similarity transform. Consider figure 5.5. The three points, E_1, E_2 and E_3, define the unit length and the xy-plane of our new coordinate system, $X_E Y_E Z_E$, with X_E as the origin. The normal to this plane defines the new z-axis. The task is to compute the position P with respect to this new coordinate frame.

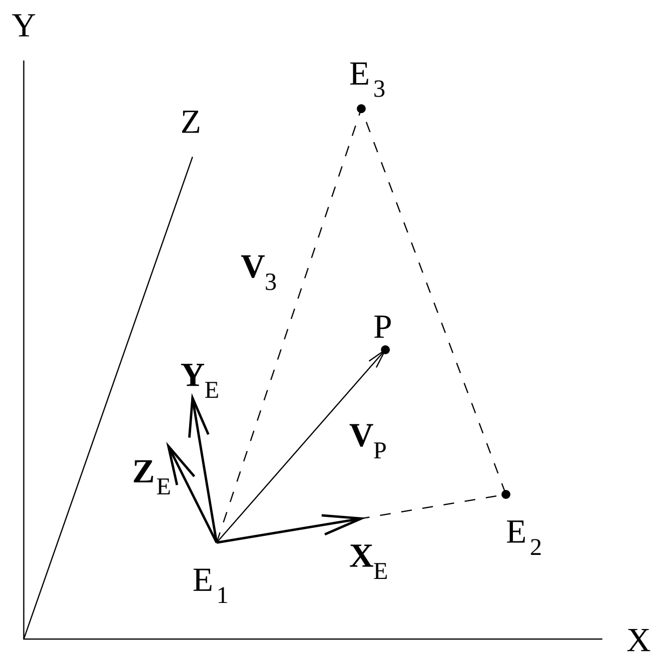


Figure 5.5: Points E_1, E_2 and E_3 define a 3-point basis for computing the new coordinates of P, which are invariant under 3D rigid and similarity transformations.

We begin by constructing orthogonal vectors corresponding to the direction of each of our new coordinate axes, as follows:

1. The new x-axis vector is simply $\mathbf{X}_E = \mathbf{E}_2 - \mathbf{E}_1$.

2. To compute the new z-axis vector we first need to define the vector $\mathbf{V}_3 = \mathbf{E}_3 - \mathbf{E}_1$. Together, the vectors $\mathbf{X}_E$ and $\mathbf{V}_3$ define our new xy-plane. We can now obtain our new z-axis as the normal to this plane, given by the cross-product $\mathbf{Z}_E = \mathbf{X}_E \times \mathbf{V}_3$.

3. Finally, our new y-axis vector is given by the cross-product $\mathbf{Y}_E = \mathbf{Z}_E \times \mathbf{X}_E$.

The invariant coordinates of the point P, denoted $\mathbf{P}_{inv} = (\alpha, \beta, \gamma)$, are obtained by calculating the component of the vector $\mathbf{V}_P = \mathbf{P} - \mathbf{E}_1$, in the direction of each of our new coordinate axis vectors, and dividing by the unit length, as follows:

1. $\alpha = (\mathbf{V}_P \cdot \mathbf{X}_E / |\mathbf{X}_E|)/|\mathbf{X}_E|$

2. $\beta = (\mathbf{V}_P \cdot \mathbf{Y}_E / |\mathbf{Y}_E|)/|\mathbf{X}_E|$

3. $\gamma = (\mathbf{V}_P \cdot \mathbf{Z}_E / |\mathbf{Z}_E|)/|\mathbf{X}_E|$

As a quick check that this method actually works, consider computing invariant coordinates from a set of four 3D points under some simple transformations. In each case, the first three points are used to compute the invariant coordinates of the fourth. Firstly, the results for the original four points.

$$\mathbf{P}_1 = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} 20 \\ 12 \\ 12 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} 14 \\ 20 \\ 15 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} 15 \\ 11 \\ 25 \end{pmatrix} \tag{5.6}$$

These points give the invariant coordinates of $\mathbf{P}_4$ as $(\alpha, \beta, \gamma) = (0.76, 0.51, 1.22)$ and a unit length of 10.39.

After scaling the points by a factor of two:

$$\mathbf{P}_1 = \begin{pmatrix} 20 \\ 20 \\ 20 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} 40 \\ 24 \\ 24 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} 28 \\ 40 \\ 30 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} 30 \\ 22 \\ 50 \end{pmatrix} \tag{5.7}$$

As one would expect, this time the unit length has increased to 20.78, but the invariant coordinates of $\mathbf{P}_4$ remain as $(0.76, 0.51, 1.22)$.

This time a translation of $(5, 10, 20)^T$, followed by the same scaling as above, transforms the points to:

$$\mathbf{P}_1 = \begin{pmatrix} 30 \\ 40 \\ 60 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} 50 \\ 44 \\ 64 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} 38 \\ 60 \\ 70 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} 40 \\ 42 \\ 90 \end{pmatrix} \tag{5.8}$$

Which gives identical results to the previous test of unit length 20.78 and invariant coordinates $(0.76, 0.51, 1.22)$.

For one quick final test, the points were transformed by a rotation of 180 degrees about the z-axis, followed by a rotation of 90 degrees about the x-axis, giving:

$$\mathbf{P}_1 = \begin{pmatrix} -10 \\ -10 \\ 10 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} -20 \\ -12 \\ 12 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} -14 \\ -15 \\ 20 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} -15 \\ -25 \\ 11 \end{pmatrix} \tag{5.9}$$

Once again, these give the invariant coordinates of $\mathbf{P}_4$ as $(0.76, 0.51, 1.22)$ and a unit length of 10.39.

5.3.2 Symmetry Considerations

For any given set of three points, A, B and C, there are six different ways of ordering them to construct a Euclidean basis: ABC, ACB, BAC, BCA, CAB and CBA. Each of these bases can be used to compute the invariant coordinates of a fourth point $D = (\alpha_i, \beta_i, \gamma_i)$, where $i = 1 \ldots 6$. An interesting question is whether or not one would expect to obtain different invariant coordinates with each basis, or if it is possible to extrapolate the result of one computation from another and thus speed up the hashing process.

Let A_1/B_1 and A_2/B_2 be two different labellings for the points A and B corresponding to the normal ordering of points (ABC) and when the positions of A and B have been swapped

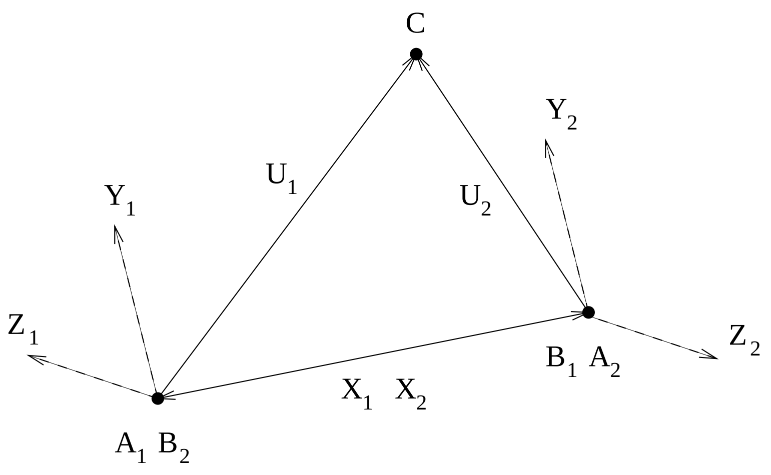


Figure 5.6: Effect of basis point ordering on coordinate axes.

(BAC), respectively. Figure 5.6 shows the two bases A_1B_1C and B_2A_2C. As described in section 5.3.1, the first stage in the computation of invariant coordinates is the construction of a new set of coordinate axes. The x-axis vectors for the two bases are $\mathbf{X}_1 = \mathbf{B}_1 - \mathbf{A}_1$ and $\mathbf{X}_2 = \mathbf{B}_2 - \mathbf{A}_2$, which differ only in their signs. The next step is to compute the vectors $\mathbf{U}_1 = \mathbf{C} - \mathbf{A}_1$ and $\mathbf{U}_2 = \mathbf{C} - \mathbf{A}_2$. Due to the change of point ordering, these two vectors will be different. The z-axes are computed as the cross-products $\mathbf{Z}_1 = \mathbf{X}_1 \times \mathbf{U}_1$ and $\mathbf{Z}_2 = \mathbf{X}_2 \times \mathbf{U}_2$. Thus $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are both normal to the plane ABC, but have different signs. It follows, that the y-axes, which are computed as the cross-products of the x and z-axes vectors, will be the same for both bases. [2]

Now, consider figure 5.7 and the addition of a fourth point D. The invariant coordinates of D are obtained by projecting the $\mathbf{V}_1$ and $\mathbf{V}_2$ vectors onto the coordinates axes defined by the two bases. First of all, it can be seen that the two vectors $\mathbf{V}_1$ and $\mathbf{V}_2$ are related as follows:

$$\mathbf{V}_1 - \mathbf{V}_2 = \mathbf{X}_1 \tag{5.10}$$

Thus, projecting in the direction of the $\mathbf{X}_1$, gives a relation between the invariant α-coordinates for the point D in the two bases:

$$\alpha_1 + \alpha_2 = 1 \tag{5.11}$$

[2]Since $\mathbf{i} \times \mathbf{j} = \mathbf{k} \longrightarrow -\mathbf{i} \times -\mathbf{j} = \mathbf{k}$.

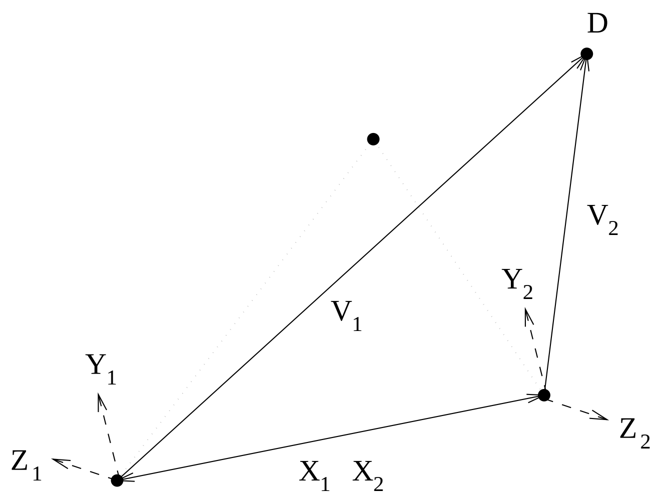


Figure 5.7: Effect of basis point ordering on invariants.

The y-axis vectors for both bases are identical, therefore the invariant β-coordinates are identical. Similarly, the z-axis vectors differ by only by their signs, therefore so do the invariant γ coordinates. As a result, if the three invariant coordinates are computed for one of these bases, then the invariants for the other are trivially defined.

This result holds for the basis pairs ABC/BAC, ACB/BCA and CAB/CBA. Thus, out of a possible total of eighteen different invariant coordinates[3], there are only nine independent values, which can be obtained from just three of the basis orderings. Table 5.1 summarises the relationships between the ordering of the basis points and the invariant coordinates.

ABC	α_1	β_1	γ_1
BAC	$1-\alpha_1$	β_1	$-\gamma_1$
ACB	α_2	β_2	γ_2
BCA	$1-\alpha_2$	β_2	$-\gamma_2$
CAB	α_3	β_3	γ_3
CBA	$1-\alpha_3$	β_3	$-\gamma_3$

Table 5.1: Effect of basis ordering on invariant coordinates.

[3]Three for each of the six different bases.

This symmetry property can be used to reduce the computation time of the hashing process, since only half of all the possible basis choices for a given point set need to be considered.

The important point is that, in each of the related basis pairs, only the positions of the A and B points have been swapped and therefore only the signs of the x-axis vectors differ. Other positional changes have different effects. For example, consider the two bases ABC and ACB, for which the positions of points B and C have been exchanged. This effectively swaps the x-axis vector $\mathbf{X}$ and the $\mathbf{U}$ vector, and changes the unit length. The z-axis vector, obtained as the cross-product of these, only changes sign[4], but the y-axis vector, which is the cross-product of the x and z-axes vectors, will be different. As a result, projecting the point vector $\mathbf{V}$ onto each of these axes and dividing by the unit length, will give different invariant coordinates for D in the two bases.

5.4 Geometric Hashing and Image Sequences

A geometric hashing system has been developed, based on the Euclidean invariants discussed in the previous section. In this section we describe how it has been incorporated into our incremental reconstruction system.

The most important difference between our method and that of standard geometric hashing, is that we *do not perform a preprocessing step*, at least, not in the usual sense. The reason for this is simple: there is nothing to preprocess, no predefined database of models to be recognised, no *a priori* knowledge of the scene. Rather, the intention is that the system should acquire its own models automatically and do so concurrently with the matching stage.

Each step of the reconstruction system processes the next pair of images in the sequence and, starting with a set of matched points, eventually recovers a 3D Euclidean structure segment for the part of the scene currently being viewed. These segments are stitched together, incrementally, to build up a reconstruction for the entire scene.

[4] In fact, since no matter which ordering of points is chosen, the resultant $\mathbf{X}$ and $\mathbf{U}$ vectors always lie in the plane, ABC, the z-axis vector is the same (up to a change of sign) for all these basis permutations.

The model acquisition part of the geometric hashing system follows along the same lines. As the reconstruction system generates new structure segments, their 3D points are used to compute Euclidean invariants (section 5.3.1) with which to index and update the, initially empty, hash table (section 5.2.2). Thus, a complete hash table representation for the entire scene is gradually obtained.

Similarly, in the recognition stage, recovered structure segments are used to compute invariants with which to index the hash table and tally votes (section 5.2.3). Hence, if a structure segment containing points that have previously been hashed comes back into view later in the sequence, the system should recognise this fact, in addition to providing the transformation between the corresponding points.

5.4.1 Partitioning the Hash Table

One approach to updating the hash table with new structure information would be to hash the new 3D points together with all of the old ones, which have previously been estimated. This would create one large model of the scene viewed over the whole image sequence, but it would be a highly redundant representation, since the majority of combinations of points used in computing the invariants could never actually be viewed together in the scene.

The hashing strategy actually used in the implementation of this scheme is based around a simple observation: both the model acquisition and recognition stages depend entirely upon the recovered structure segments *which contain only local information*. In other words, for a long image sequence, each image only gives a view of a small section of the entire scene. Thus, a more efficient approach to updating the hash table would be to compute invariants using only the currently viewed structure segment. In fact, the method used is to compute invariants based on points in the current and previous d structure segments[5]. This creates some overlap in the hash table, but allows for recognition from a wider range of viewpoints.

In essence, when a segment of scene structure is viewed for the first time in the image sequence, it is used, along with neighbouring segments, to construct an invariant representation for

[5]Where d is determined by the disparity between pairs of images in the sequence.

the local patch of structure. Thus, when updating the hash table, the information recorded is not (basis,model), but (basis,patch).

5.4.2 Algorithm Outline

Each step in the reconstruction algorithm results in a structure segment containing a set of newly estimated Euclidean points. At the same time, a set of previously estimated old structure points will have just gone out of view. The old points are used to update the hash table, while the new ones are used for recognition. The benefit of this approach is to ensure that the new points are not used for model representation and matching at the same step.

Suppose, at some step i of the reconstruction process, a structure segment has been recovered and sets of old and new points, denoted S_{OLD_i} and S_{NEW_i}, have been determined. Model acquisition and matching proceed as follows:

Model Acquisition

Combine the points in S_{OLD_i} with those obtained at d previous stages ($S_{OLD_{i-1}} \ldots S_{OLD_{i-d}}$), to obtain a local structure *patch*. Let this patch contain a total of m points. For each ordered, non-collinear triplet of these points: (Euclidean basis):

1. Compute the invariant coordinates (α, β, γ) of the remaining $m - 3$ points points in the Euclidean coordinate frame defined by the basis triplet.
2. Use the invariants to generate indices into a 3D hash table.
3. At each given hash table location, store a record of the patch and Euclidean basis from which the invariants were obtained.

Matching

1. Suppose S_{NEW_i} contains a total of n points.

2. Choose an arbitrary ordered triplet of non-collinear points as a Euclidean basis and compute the invariant coordinates of the remaining $n - 3$ points.

3. Use the invariants to generate indices into a 3D hash table.

4. Check each indexed hash table bucket and tally a vote for every (patch,basis) pair stored there.

5. Look for a (patch,basis) pair which scores a high number of votes. Each such pair implies that at least part of the given patch is present in the scene. The uniquely defined Euclidean transformation between the *candidate* patch and scene segment bases is assumed to be the transformation that maps between the patch and the scene.

6. Apply the transformation obtained in step 5 to all the S_{NEW_i} points that voted for the candidate patch, to induce additional point correspondences. Find the best transformation between all the correspondences, in a least-squares sense.

7. Compute the average Euclidean distance between the candidate patch points and those obtained by applying the above transform to S_{NEW_i}.

8. If the transformation results in a low average Euclidean distance between points in S_{NEW_i} and the candidate patch, then the existence and orientation of the patch has been verified and matching is complete. If not, this candidate solution is rejected and another one from step 5 is examined. If there are no more candidate solutions, go back to step 2.

5.5 Results

5.5.1 Synthetic Data: Simple Models

We begin with a very basic test of the geometric hashing system, to ensure that the implementation is algorithmically correct and bug-free.

Figure 5.8 shows a set of simple 3D models, the first five of these are quite distinct, while the sixth is a slightly skewed (noisy) version of the first. The corner points of models 1–5 were used

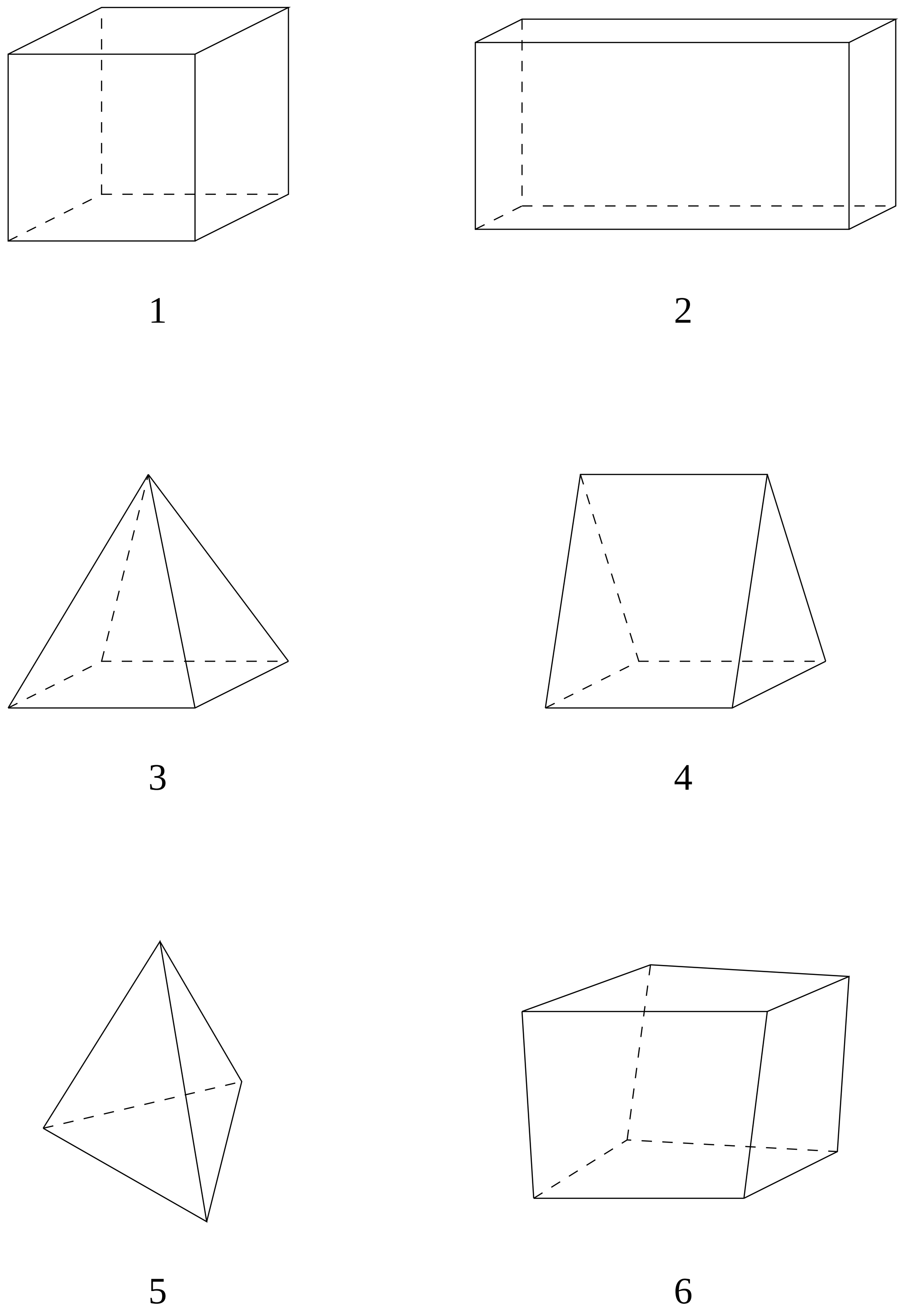


Figure 5.8: The set of simple models used in initial tests on the geometric hashing system. Models 1–5 are structure segments used to build the hash table structure. Model 6 is a skewed version of model 1, used to see if the system copes with noisy data.

as structure patches, with which to initialise the hash table, as described in the previous section. Subsets of the *same* points were then fed into the matching process, to see if the system could recognise each of the models using perfect data. For this test, an exhaustive[6] series of trials were performed, with the best match taken to be the model which received the most votes in any single trial. Results are shown in Table 5.2.

Subset Model	Matched Model	Most Votes	Trial
1	1	5	1
2	2	5	1
3	3	2	1
4	4	3	1
5	5	1	1

Table 5.2: Results of geometric hashing using perfect synthetic data.

As one would expect, the system had no difficulty matching the point data to the original models. In every case the correct match, with the highest number of votes, was achieved with the first choice of basis points. It is worth noting that each of the 'Most Votes' values is actually the maximum number of votes possible, since an n-point structure patch will cast $n - 3$ votes per basis[7].

We mention, in passing, that exactly the same results as above were obtained from a supplementary test, which attempted to match using model points that had undergone arbitrary Euclidean transformations. This is additional confirmation of the invariance property discussed in section 5.3.1.

The next test of the system was an attempt matching with imperfect data. The sensitivity of the geometric hashing paradigm to such errors was discussed in [20], while methods for dealing with noisy data were presented in [70] and [16]. The suggested approach is to modify the voting part of the matching scheme. Rather than just indexing a single hash table bucket and tallying a

[6]Using all possible basis/point combinations.

[7]Assuming, as in this case, an even distribution of basis/model pairs over the hash table, such that no pair appears more than once in a given bucket.

vote for each of the elements it contains, all buckets within a *region of interest* around the indexed location take part in the voting process. The implementation of this idea required only a minor extension to our 3D geometric hashing system, whereby votes are cast in a spherical region around the indexed location.

The imperfect data we hoped to match was based on the sixth model of figure 5.8. In fact, several variations of this model were created by adding different levels of random noise [8] to the original model 1. Results from a series of trials, as before, are shown in Table 5.3.

% Noise	Matched Model	Most Votes	% Max Correct
0	1	5	100
5	1	4	100
10	1	4	100
15	1	3	100
20	1	2	90
30	1	1	66
40	3	1	46

Table 5.3: Results of geometric hashing using noisy synthetic data

The first thing to note from the table is that up to and including the 30% noise level the geometric hashing system successfully matched the input data with model 1. However, as the noise level increased, so the largest number of votes scored in any single trial decreased. The final column of the table gives the percentage of the trials with the highest vote count that proposed the correct match. In the 10–15% noise range this was all of them. At the 20% noise level this dropped to 90%, and with 30% noise only 66% of trials with the given highest vote count (1) matched correctly. This means that although the matching process was successful eventually, many more candidate matches had to go through the verification procedure. Unsurprisingly, with 40% noise the matching system proposed the wrong match the majority of the time.

[8] The noise level was calculated as a percentage of the size of the original model. Model 1 was a cube of $10\times10\times10$ units, so, for example, a noise level of 10% means a random coordinate shift of ±1 unit.

5.5.2 Real Data: Recovered Structure

For this series of tests we attempted to match the structure recovered from the first pair of quadrangle images, against the original ground truth. The structure correspondences were grouped into five models/patches according to 3D location, as shown in Table 5.4.

Model Number	Number of Points	Description of features
1	8	Supporting pillars at the front of the lecture theatre
2	9	Supporting pillars at the side of the lecture theatre
3	6	Features on the distant building
4	14	Windows on the left of the lecture theatre
5	13	Windows on the right of the lecture theatre

Table 5.4: Grouping of features into structure patches

Each of the recovered structure patches was used as input to the matching process. Initial experiments with this data highlighted a problem. The system was proposing (incorrect) candidate matches with a 'Most Votes' value that was three or four times the expected maximum. This was caused by an uneven distribution of invariant coordinate values, as shown in Figure 5.9. As a result, a given basis/model pair could be stored many times in the same hash table bucket. Thus, each time that bucket was indexed for voting, multiple votes were cast for that pair. We attempted to solve this problem using a logarithm-based hash function.

The results of subsequent experiments are shown in Table 5.5. Unfortunately, none of the recovered structure patches were successfully matched against the corresponding ground truth data. The 'Most Votes' column still shows a larger number of votes for the proposed match which is bigger than the maximum we would expect from an evenly distributed hash table, but there has been a marked improvement. The final column gives the largest number of votes cast in a single trial for the *correct* model. Only the trials with patches 4 and 5 result in a high number of votes for the correct model. In the other trials the proposed best match seems almost arbitrary.

This is not really surprising if we take a closer look at the data the geometric hashing system

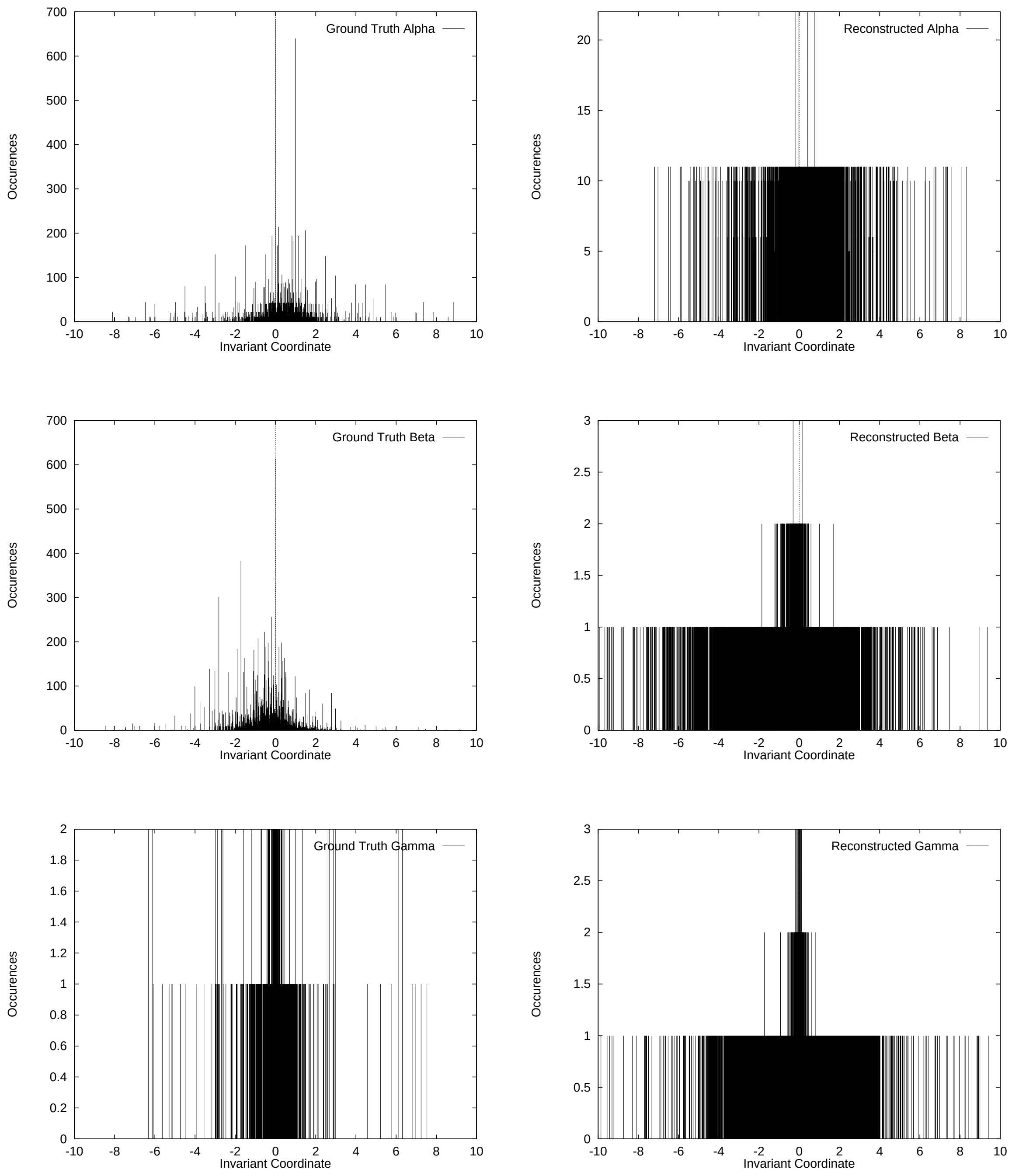


Figure 5.9: Distributions of invariants. The left column of graphs shows the distribution of α, β and γ invariants for ground truth data. The right hand column shows the distributions for the corresponding recovered structure. The ground truth graph for γ has had the zero entries removed, as they were swamping the other values. This was caused by many of the basis/point pairs being coplanar.

Recovered Patch	Matched Model	Most Votes	Correct Votes
1	4	12	0
2	5	13	7
3	4	5	0
4	1	17	14
5	4	16	15

Table 5.5: Results of geometric hashing using real data.

has to work with. For example, four points chosen at random from set of correspondences number 2. The ground truth values for these points are:

$$\mathbf{P}_1 = \begin{pmatrix} 20.00 \\ 39.00 \\ -350.00 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} 0.00 \\ 39.00 \\ -743.00 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} -662.00 \\ 108.00 \\ 244.00 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} -587.00 \\ 328.00 \\ 1050.00 \end{pmatrix} \tag{5.12}$$

The invariant coordinates of $\mathbf{P}_4$ are $(\alpha, \beta, \gamma) = (-3.47, -1.78, 0.56)$, using the first three points as a basis. If we now examine the corresponding recovered structure points:

$$\mathbf{P}_1 = \begin{pmatrix} -33.46 \\ 26.53 \\ -304.85 \end{pmatrix}; \mathbf{P}_2 = \begin{pmatrix} -75.79 \\ 28.32 \\ -688.49 \end{pmatrix}; \mathbf{P}_3 = \begin{pmatrix} -731.63 \\ 99.23 \\ 331.68 \end{pmatrix}; \mathbf{P}_4 = \begin{pmatrix} -535.36 \\ 325.74 \\ 935.04 \end{pmatrix} \tag{5.13}$$

These give the invariant coordinates of $\mathbf{P}_4$ as $(\alpha, \beta, \gamma) = (-3.05, -1.71, 0.62)$. The β and γ values are quite similar, but in the context of the overall range, the α value has changed quite dramatically. Such differences between the computed invariants cause the wrong hash table location to be indexed, resulting in spurious votes being cast and incorrect candidate matches.

5.6 Conclusions

Although the results of experiments with synthetic data have been encouraging, so far we have been unable to get the geometric hashing system to recognise real structure patches successfully. It is apparent that the cause of this problem is not the geometric hashing system itself, but rather, inaccuracies in the recovered Euclidean structure, which forms the input to the system.

A common approach to dealing with such noisy data is to use a region-based voting algorithm, but this has also proved inadequate. The difficulty lies in determining the size of the region of interest. Too small a region achieves nothing, but if it is too large then many candidate matches are found, which defeats the object of performing geometric hashing in the first place. So far we have been unable to develop a sensible value for the size of the voting region which is large enough to allow recognition to take place but small enough to generate a manageable number of candidate matches.

It may be that performance can be improved slightly by careful choice of parameters such as the size of the voting region or modifying the hash function, but, in conclusion, it would seem that unless a way can be found to produce more accurate structure estimates, the geometric hashing method of matching will not be successful.

Chapter 6

Conclusions

6.1 Summary

The work presented in this thesis addresses the problem of automatically reconstructing a model of an extended environment, from a long image sequence taken with an ordinary hand-held video camera. An uncalibrated approach to structure recovery has been taken, based on the calculation of the epipolar geometry of successive pairs of images in the sequence. Knowledge of ground truth data, for the first image pair *only*, is used to propagate an estimate for the Euclidean structure of the entire scene. Over a long image sequence, it is anticipated that errors will accrue, and as a result, the recovered structure will be internally inconsistent. A method which attempts to detect such anomalies has been developed, using the geometric hashing paradigm.

In Chapter 3 a method is described for acquiring the fundamental matrix from a set of matched points in a pair of uncalibrated images. It is based on a novel implementation of the well-known 8-point algorithm, which combines a recently developed normalisation technique with a variation on the RANSAC parameter estimation algorithm. This linear method is simple, yet efficient, and experiments on a range of real and synthetic data show that it produces quantifiably accurate results. A direct comparison demonstrates that the method generates fundamental matrix estimates of similar quality to those of more complicated alternatives.

The fundamental matrix is the input to the reconstruction system presented in Chapter 4.

We use Hartley's method for factorising a fundamental matrix, to construct a pair of camera matrices for the corresponding images. These are used to obtain an estimate for the projective structure of the current view, via back-projection. For the first pair of images in the sequence, five or more ground truth points are required, in order to compute a projective transform matrix with which to upgrade from projective to Euclidean structure. The problem of calculating the projectivity which best fits the data is again formulated and solved using the RANSAC scheme.

An original feature of the reconstruction system is the way in which remaining image sequence pairs are processed. Each is used to obtain a segment of projective structure, as outlined above, but the upgrade to Euclidean is performed, not with ground truth data, but rather, using previously estimated Euclidean structure. Thus a complete reconstruction for the scene is incrementally obtained, by stitching together the segments of Euclidean structure acquired at each step. Results show that the reconstruction system performs well when recovering the structure of objects which are close to the camera, but copes poorly with distant objects. In the latter case, the image resolution is insufficient to be able to detect and match point features accurately, even by hand. As a result of this, very small changes in the image coordinates of a point can lead to large changes in the recovered structure. This is a problem faced by any image feature-based reconstruction system and is not due to the specific method used here.

At each step of the reconstruction system, new structure is acquired using old structure that was itself estimated. As successive image pairs are processed the error in the recovered structure will accumulate. In a long image sequence there is the possibility that previously viewed parts of the scene will be re-encountered. In this event there will be two structure estimates for the scene segment; the current one and the one obtained the first time it was in view. It is required that a match between these two segments is found and their relative orientation obtained, in order that globally consistent structure can be maintained. Chapter 5 describes an method to tackle this problem, using geometric hashing.

The geometric hashing system has been developed to work in tandem with the reconstruction process. Unlike conventional geometric hashing, there is no off-line preprocessing stage. Rather, the models to be recognised are *memorised* while the system is running, and this is done concurrently with the matching stage. The models in question are patches of recovered Euclidean structure. At each step in the reconstruction, an additional patch of structure is used to update the hash table. The system is based around 3D Euclidean invariants and a method for obtaining these is described. Also presented is a symmetry-based technique for improving the efficiency of the hashing process.

The goal of the geometric hashing system is not to produce a unique match between the scene structure and that which is stored in the hash table. Instead, it aims to obtain a manageable number of candidate matches, which can be examined more closely. Unfortunately, this aim has not yet been achieved, as the system either produces large numbers of candidate matches, or not at all. This failure is not due to the geometric hashing implementation, but rather, the data it works with. The problem is caused by the fact that errors in the recovered Euclidean structure accumulate more quickly than expected. For the geometric hashing system to function as intended, the accuracy of the reconstruction must be improved.

6.2 Future Work

The calculation of the projective transform between projective and Euclidean structure is the key to the reconstruction process. The elements of the transform matrix are obtained via the solution of a set of homogeneous linear equations, using the least eigenvector method. This is the same approach taken when computing the fundamental matrix in Chapter 3, and it would be worth investigating the possibility of improving the result of this calculation by developing a normalising transform, based on the analysis in [26].

Another extension to the existing method might be to assign a measure of accuracy to each reconstructed point, and only use the most accurate points in the projectivity calculation. The measure might be based on the distance of the point from the camera, or recursively, on the accuracy of the points used when its structure was recovered.

A possible cause of structure recovery problems is the initial reliance on ground truth data. Any measurement error in these points would have an adverse effect on the whole reconstruction. An alternative approach would be to estimate the Euclidean structure directly, using a self-calibration, as in [25].

In its present form the geometric hashing system is is based on invariants of a 3D similarity or rigid transform. However, since we *are* dealing with rigid structures it is possible to obtain more discriminatory information, for example the size of the triangle formed by the three basis points. This *shape signature* can be used to ensure that only appropriate bases receive votes and thus reduce the number of candidate matches [70].

Also, at present, there is no indication of *when* the recognition process should take place. Of course, it is possible to attempt recognition after every reconstruction step and hash table update, but this would be wasteful, since it is expected that previously viewed structure will only rarely be re-encountered. The reconstruction process can provide an estimate for the pose of the camera at each step and this odometry information could be used to trigger recognition when approaching structure that has been seen before.

In addition, once a match has been found, there is still the question of how best to update the structure, so that it internally consistent. This is of particular importance if the reconstruction is to be of practical use, for example a virtual reality application, and is certain to be the subject of future research.

6.3 Closing Comments

The problem we have tackled is an exceptionally difficult one, and the proposed solution brings together elements from a number of areas of machine vision. To date, we have been unable to demonstrate successful resolution of inconsistencies in the reconstructed model, for reasons discussed in Chapter 5. However, the overall approach proposed remains plausible and it is hoped that future work, including some of the suggestions above, will succeed in producing internally consistent models.

Appendix A

Essential Projective Geometry

In this appendix we review some of the most important concepts of projective geometry, which we have used elsewhere in the thesis. For a more thorough introduction, the reader is referred to [11] or the appendix of [47], which provide excellent discussions of the subject, from a machine vision standpoint.

A.1 Homogeneous Coordinates

In projective geometry manipulation of points, lines, planes etc. is carried out using homogeneous coordinates. Projective transformations are linear in homogeneous coordinates, and some problems can be greatly simplified by expressing them in this manner. Consider the case of perspective projection from 3D to 2D, which is important in machine vision as it represents the formation of an image by a camera. Using Cartesian coordinates this transformation is non-linear, but it is linear in homogeneous coordinates [11].

In two dimensions, the Cartesian coordinates of a point are a 2-vector $(x, y)^T$. The same point in projective two-space, $\mathcal{P}^2$, can be represented in homogeneous coordinates by some 3-vector $(u, v, w)^T$. The simplest way to obtain the values of these three elements is to set $u = x, v = y$ and $w = 1$, thus:

$$(x, y)^T \rightarrow (x, y, 1)^T \tag{A.1}$$

However, an important property of projective geometry is that only the ratios of the elements of the homogeneous coordinates are important. Hence two homogeneous vectors represent the same homogeneous point if one is a multiple of the other. That is:

$$(x, y, 1)^T \equiv (\lambda x, \lambda y, \lambda)^T \tag{A.2}$$

where λ is some non-zero scalar. For example, $(12, 10, 2)^T, (30, 25, 5)^T$ and $(-6, -5, -1)^T$ all represent the same homogeneous point.

Converting from homogeneous back to Cartesian coordinates is equally straightforward. Simply divide through the homogeneous vector by its third element and then remove the third element (which will of course be 1),

$$(\lambda x, \lambda y, \lambda)^T \rightarrow \left(\frac{\lambda x}{\lambda}, \frac{\lambda y}{\lambda}, \frac{\lambda}{\lambda}\right)^T \rightarrow (x, y, 1)^T = (x, y)^T \tag{A.3}$$

Or, without the explicit scale factor, $(u, v, w)^T \rightarrow (u/w, v/w)^T$. Thus, in our example above, the three homogeneous vectors all correspond to the Cartesian point $(6, 5)^T$.

An additional benefit of homogeneous coordinates is that they make it possible to represent points located at infinity on the image plane. There is no such representation in Cartesian coordinates. In homogeneous coordinates a point at infinity, called an *ideal point* has its third element equal to zero i.e. it is of the form $(u, v, 0)^T$. Ideal points are treated in exactly the same way as any other points in the image plane. The set of all ideal and non-ideal points in projective 2-space is called the *projective plane* and is denoted $\mathcal{P}^2$.

The equation of a line in two dimensions can be expressed in Cartesian coordinates as

$$ax + by + c = 0 \tag{A.4}$$

which can be rewritten in terms of homogeneous coordinates as

$$a\frac{u}{w} + b\frac{v}{w} + c = 0 \leftrightarrow au + bv + cw = 0 \tag{A.5}$$

or, using the vector dot product

$$\mathbf{l}.\mathbf{p} = \mathbf{0} \tag{A.6}$$

where $l = (a, b, c)^T$ is the homogeneous 3-vector representation of the line. As with points, only the ratios of the three elements are important, as we can see that multiplying equation A.5 by a scalar has no effect.

A.2 Some Simple Constructions

A.2.1 Computing The Line Through Two Points

From equation A.5 we can see that for a line $\mathbf{l} = (a, b, c)^T$ to pass through the two points $\mathbf{p} = (p_u, p_v, p_w)^T$ and $\mathbf{q} = (q_u, q_v, q_w)^T$, the following relations must be satisfied

$$ap_u + bp_v + cp_w = 0 \qquad \text{and} \qquad aq_u + bq_v + cq_w = 0 \tag{A.7}$$

The solution to these equations can be obtained from the cross product

$$\mathbf{l} = \mathbf{p} \times \mathbf{q} \tag{A.8}$$

We can see that this is so by noting that if a point $\mathbf{p}$ lies on a line $\mathbf{l}$, the dot product of their two coordinate vectors is zero (equation A.5). From the properties of the vector triple product we get $\mathbf{p}.\mathbf{l} = \mathbf{p}.(\mathbf{p} \times \mathbf{q}) = \mathbf{0}$, and we obtain the result above[1] .

[1] Since $\mathbf{p}.(\mathbf{p} \times \mathbf{q}) = (\mathbf{p} \times \mathbf{p}).\mathbf{q}$ and $\mathbf{p} \times \mathbf{p} = \mathbf{0}$.

A.2.2 The Intersection of Two Lines

This is the dual of the previous problem and we obtain similar constraints in that the point of intersection $\mathbf{p} = (u, v, w)^T$ of the two lines $\mathbf{l} = (l_a, l_b, l_c)^T$ and $\mathbf{m} = (m_a, m_b, m_c)^T$, must satisfy

$$l_a u + l_b v + l_c w = 0 \qquad \text{and} \qquad m_a u + m_q v + m_q w = 0 \tag{A.9}$$

This similarity highlights the so-called *principle of duality*, which states that for any manipulation involving projective points and lines, each point can be exchanged for a line and each line can be exchanged for a point, the result being the dual manipulation to the original. Thus, the solution to this problem can be expressed simply as

$$\mathbf{p} = \mathbf{l} \times \mathbf{m} \tag{A.10}$$

A.2.3 Normalising Homogeneous Coordinates

When carrying out lots of projective geometry calculations, the homogeneous coordinates of points and lines may become very small or large. Therefore, it is a good idea to normalise the homogeneous coordinates at each stage in the computation, to avoid numerical error. As a homogeneous vector can be multiplied by any scalar and still represent the same point or line, normalisation can be done quite simply. For a point $\mathbf{p} = (u, v, w)$, we use the following normalisation:

$$(u, v, w)^T \rightarrow \left(\frac{u}{w}, \frac{v}{w}, 1\right)^T \tag{A.11}$$

and for the line $\mathbf{l} = (a, b, c)$

$$(a, b, c)^T \rightarrow \left(\frac{a}{\sqrt{(a^2 + b^2)}}, \frac{b}{\sqrt{(a^2 + b^2)}}, \frac{c}{\sqrt{(a^2 + b^2)}}\right)^T \tag{A.12}$$

A.2.4 The Perpendicular Distance of a Point from a Line

The normalisation of points and lines described above, leads to a very simple method for computing this quantity. Specifically, the perpendicular distance d of a point p from a line l is given by the dot product:

$$d = \mathbf{p}.\mathbf{l} \tag{A.13}$$

So in the case where the point and line are incident we get the anticipated result:

$$\mathbf{p}.\mathbf{l} = 0 \tag{A.14}$$

A.3 Projective Transformations

Transformations within and between projective spaces are called *projectivities*. In mathematical studies of projective geometry there is no emphasis on projective space of any particular dimension, but in computer vision, some cases are more interesting than others.

The previous sections have concentrated on the manipulation of points and lines in $\mathcal{P}^2$, which have been widely used in our work on the fundamental matrix (chapter 3). Projective 3-space, $\mathcal{P}^3$, is a generalisation of the projective plane, following from the definition of $\mathcal{P}^2$ in section A.1. Again, homogeneous coordinates are used, here with all their dimensions increased by one. That is, a point in $\mathcal{P}^3$ is represented by the homogeneous 4-vector $(X, Y, Z, T)^T$. In the remainder of this section we will briefly discuss two projective transformations of $\mathcal{P}^3$.

A.3.1 $\mathcal{P}^3$ to $\mathcal{P}^3$

A projectivity from $\mathcal{P}^3$ to $\mathcal{P}^3$ acts on and generates a homogeneous 4-vector. It is can therefore be represented by the (non-singular) 4×4 matrix $\mathbf{H}$

$$\lambda \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.15}$$

Note that the only effect of multiplying $\mathbf{H}$ by some non-zero scalar, is to change the value of the scale factor λ on the left-hand side. However, as we have seen, all values of λ still refer to the same homogeneous point. Thus $\mathbf{H}$ is defined only up to a scale factor and has only 15 degrees of freedom.

Multiplying out and eliminating the unknown scale factor, λ, leaves three equations in the elements of $\mathbf{H}$. Thus given five corresponding points in $\mathcal{P}^3$, a system of linear equations can be formed and solved for the unknown elements of $\mathbf{H}$. If more than five point matches are known, a least-squares solution can be obtained.

A.3.2 $\mathcal{P}^3$ to $\mathcal{P}^2$

Projection from a projective space to one of lower dimensionality can be achieved by simply eliminating one of the coordinates of the transformed projective space. For example, projection from $\mathcal{P}^3$ to $\mathcal{P}^2$ can be performed by the 3×4 matrix M as follows:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} m_{11} & m_{12} & h_{13} & m_{14} \\ m_{21} & m_{22} & h_{23} & m_{24} \\ m_{31} & m_{32} & h_{33} & m_{34} \\ m_{41} & m_{42} & h_{43} & m_{44} \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.16}$$

Once again, the overall scale of the matrix M is unimportant. Hence, this transformation has 11 essential parameters. Multiplying out and eliminating the scale factor, as before, leads to a pair of equations in the unknowns. Thus, six or more 3D reference points, together with their corresponding image points are sufficient to construct and solve a system of linear equations for the elements of $\mathbf{M}$.

The general projective projection matrix $\mathbf{M}$ can account for many of the geometric aspects of image formation, including the case of viewing the projection of a projection, for example, a picture of a picture or the shape of a shadow in an image. Constraints can be applied to the form of the matrix to account for the standard case of projection from 3D space onto an image plane from a single point i.e. perspective projection. In this case, the perspective *camera matrix* $\mathbf{M}$ can be decomposed as:

$$\mathbf{M} = \mathbf{K}(\mathbf{R}| - \mathbf{R}\mathbf{t}) \tag{A.17}$$

where $\mathbf{K}$ is an upper triangular matrix, $\mathbf{R}$ is a 3D Euclidean rotation matrix and $\mathbf{t}$ is a translation vector. $\mathbf{K}$ represents the *intrinsic* parameters, which define the optical characteristics of the camera. $\mathbf{R}$ and $\mathbf{t}$ encode the *extrinsic* parameters, which define the transformation between the world and camera coordinate systems. This factorisation of $\mathbf{M}$ can be computed by QR-factorisation [18].

Appendix B

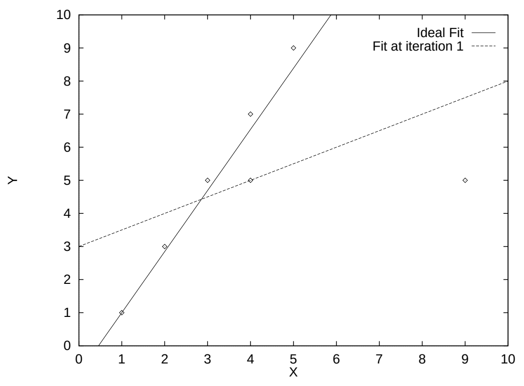
RANSAC

RANSAC (RANdom SAmple Consensus) is a paradigm for fitting a model to experimental data. It was introduced into the vision literature in 1981 by Fischler and Bolles [15], who described one possible application in the field of automated cartography. Here RANSAC was used to determine camera location from a set of 3D ground truth points and their corresponding image points. However, the general RANSAC algorithm can be applied to an unlimited number of parameter estimation and model-fitting problems.

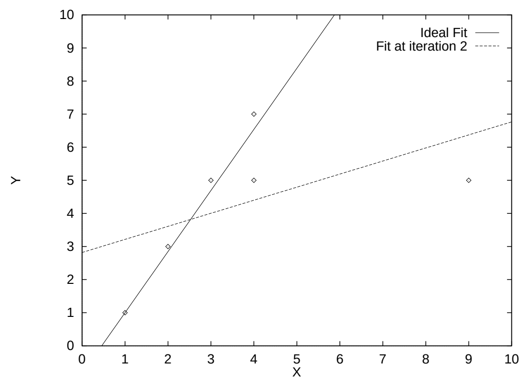
The beauty of RANSAC lies in its ability to interpret/smooth data containing a significant proportion of large errors, thus making it ideally suited to vision and image processing applications, which often rely on noisy data provided by error prone feature detectors, matchers etc. Classical parameter estimation techniques, such as least-squares, optimise the fit of a model based on all the presented data. They are averaging techniques which rely on the smoothing assumption, that the maximum error in any given data item is a function of the total size of the dataset. Hence there will always be enough accurate data items to smooth out the errors.

In many practical applications, this assumption does not hold; i.e., the dataset contains uncompensated gross errors (outliers). A common heuristic for dealing with this is firstly to use all the data to compute the model parameters, then find the data item with the highest deviation from the model fit, delete it, assuming it is an outlier, then recompute the model. This process is iterated until either the maximum deviation is less than some given threshold or there is insufficient data to continue. However, it can be seen that the presence of just one outlier can cause such a heuristic to fail.

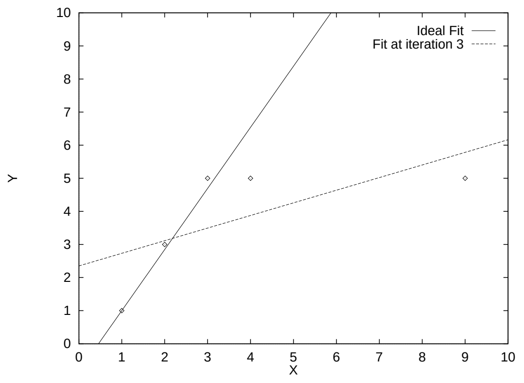
For example, consider the case of fitting a line to a set of seven 2D points. At each iteration, we compute the equation of the line and discard the point with the largest perpendicular distance to the line. Termination occurs when all remaining points lie within 1.2 units of the line. Figure B.1 shows the data sets and best fit lines computed at each of the necessary four iterations. Also shown on each graph is the ideal best fit which would be obtained if the outlier (the rightmost data point) was ignored.



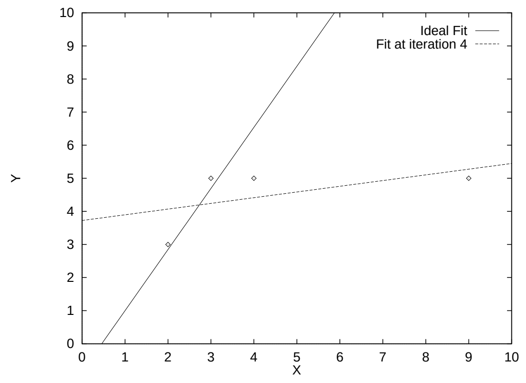
(a)



(b)



(c)



(d)

Figure B.1: Effect of an outlier on the accuracy of fitting a line to a set of points using least-squares

The tables below show which data points were used in each iteration, the parameters of the fit line and the perpendicular distance of each data point from that line.

Least Squares Approximations				
Iteration	Data Set	Fitting Line		
1	0, 1, 2, 3, 4, 5, 6	0.50x	+	3.00
2	0, 1, 2, 3, 5, 6	0.39x	+	2.81
3	0, 1, 2, 5, 6	0.38x	+	2.35
4	1, 2, 5, 6	0.17x	+	3.72

Perpendicular Distances						
Point	X	Y	Iteration 1	Iteration 2	Iteration 3	Iteration 4
0	1	1	2.23	2.05	1.61	–
1	2	3	0.89	0.56	0.10	1.05
2	3	5	0.44	0.92	1.40	0.74
3	4	7	1.78	2.41	–	–
4	5	9	3.13	–	–	–
5	4	5	0.00	0.55	1.04	0.57
6	9	5	2.23	1.27	0.73	0.27

The RANSAC procedure is the complete opposite to such traditional smoothing techniques. Instead of using as much of the dataset as possible to form an initial solution, then gradually eliminating outliers, RANSAC starts off with the smallest feasible dataset and then enlarges this with consistent data when possible. Fischler and Bolles describe the simple example of using RANSAC for fitting a circle to a set of 2D data points. Start with a three point subset, since three points are needed to define a circle. Compute the centre and radius of the candidate circle and count the number of points close enough to the circumference to suggest they belong to the circle. If there are enough such points, RANSAC would then employ a smoothing technique, such as least-squares, to obtain an improved estimate for the parameters of the circle, based on the the set of mutually consistent points.

More formally, the RANSAC paradigm can be stated as follows:

Given a model that requires a minimum of n data points to compute its free parameters and a set of data points P such that P contains more than n elements:

1. Randomly select an n-point subset S from P and use those points to instantiate the model M.

2. Obtain the subset C containing those points in P that are within some error tolerance of M. This is called the consensus set.

3. If C contains a number of points greater than some threshold t, which is a function of the number of outliers in P, then use C to compute (possibly using least-squares) a new model M'. Otherwise, randomly select another subset S and repeat the above process.

4. If, after some predetermined number of trials, no consensus set with t or more elements has been found, either terminate with an error or solve for the model using the largest consensus set that *has* been found.

5. Additional iterative steps can now be performed, if required. Once a new model M' has been computed from the consensus set, if any additional points from P are consistent with M', add them to C and recompute the model.

The original authors go on to give guidelines about the choice of the three variable RANSAC parameters: the error tolerance when deciding on a point's consistency, the threshold t and the number of subsets to try. Clearly, the value of these parameters will vary depending on the particular RANSAC application.

In fact, some applications will require modifications to the algorithm outlined above. For example, the termination condition in step 3 is dependent on a threshold value, which implies *a priori* knowledge about the number of outliers in the dataset. This is often unavailable, as in our estimation of the fundamental matrix in Chapter 3 and again in Chapter 4, when computing the transform between projective and Euclidean 3D structure. In such cases one simple solution is to abandon the threshold check and use the largest consensus set obtained after some fixed number of trials. This is the method used by Gee for a RANSAC-driven pose estimation system [17]. Another alternative for avoiding the direct dependency on the threshold value is used by Torr in [62], in a RANSAC algorithm for eliminating outliers from a set of matched points. He uses predetermined estimates for the percentage of outliers present, to calculate the number of trials required to obtain an optimum solution.

In both our RANSAC applications we have a way of obtaining a measure of the accuracy of the model instantiations. Thus an an alternative formulation of step 3, is as follows:

3. If C contains a number of points greater than or equal to the previous best consensus set then

    - Use C to compute a new model M'.
    - Obtain a measure[1] of the accuracy of M'.
    - If M' is more accurate than the best previously model, record the details of M' and C then continue.

    Otherwise, randomly select another subset S and repeat the above process.

Using this approach gives monotonically increasing consensus set sizes, each of which generates a more accurate model than the previous one. In addition, it provides a termination condition based on the accuracy of the computed models, not on the size of the consensus set. In the main text, we denote this variation on the scheme as **RANSAC***, to differentiate it from the standard method.

[1] Exactly how this is obtained is dependent on the nature of the model being computed.

Appendix C

The Quadrangle Image Sequence

This is an image sequence of a part of the campus at Leeds University, known as the quadrangle. The original video footage was digitised into around 10000 separate frames. A number of frames were selected, by hand, such that there was sufficient inter-frame overlap to be useful in our scene reconstruction system. This resulted in a sequence of just 104 images of 384$\times$288 pixels, whose thumbnails are shown on the following pages.

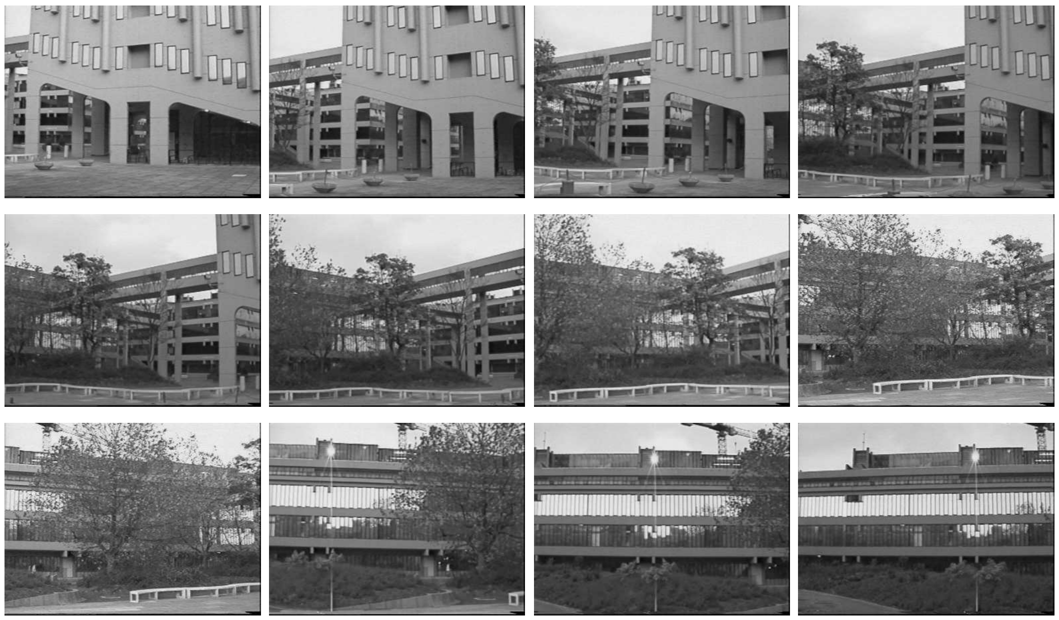


Figure C.1: Images 0 to 11

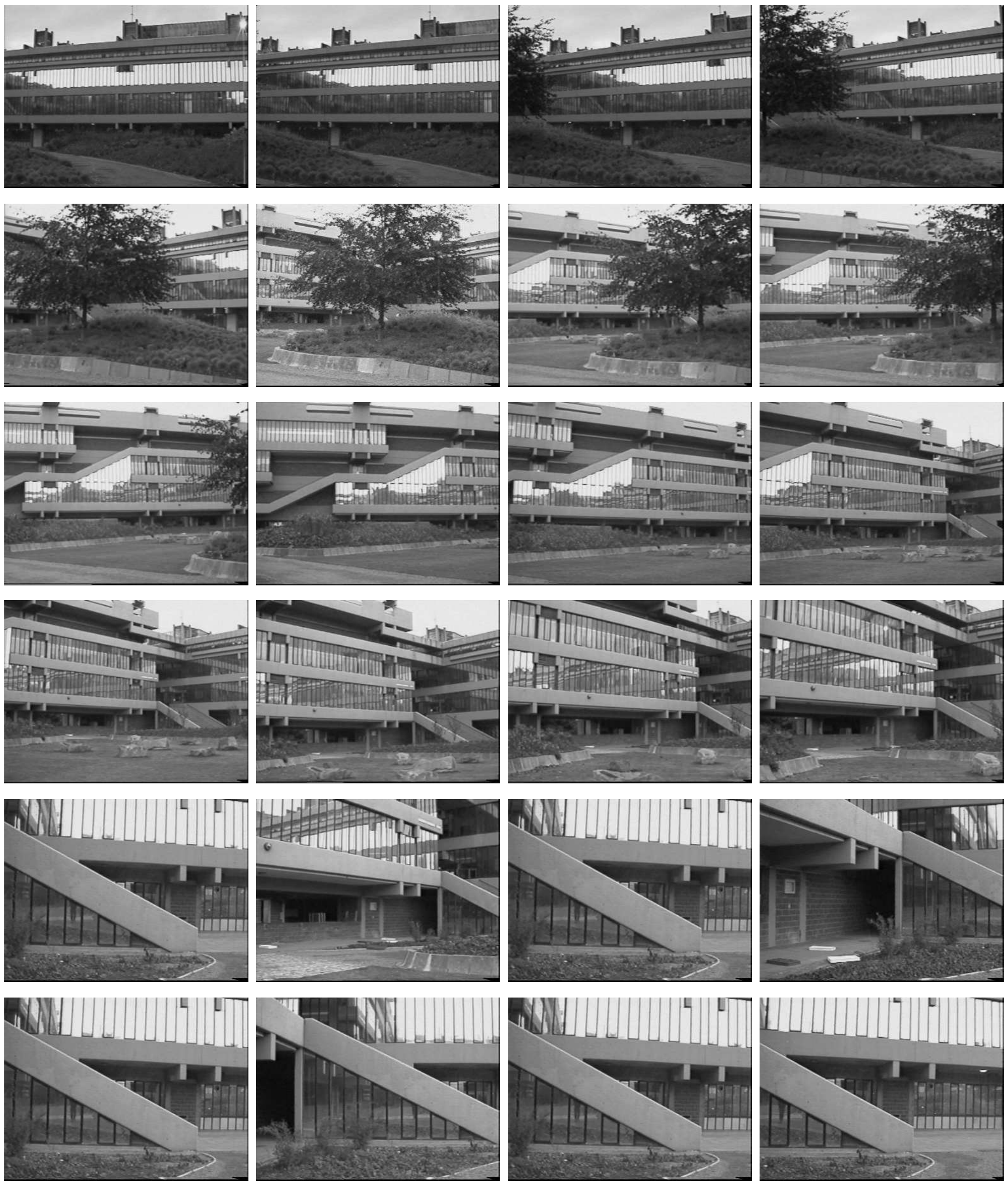


Figure C.2: Images 12 to 35

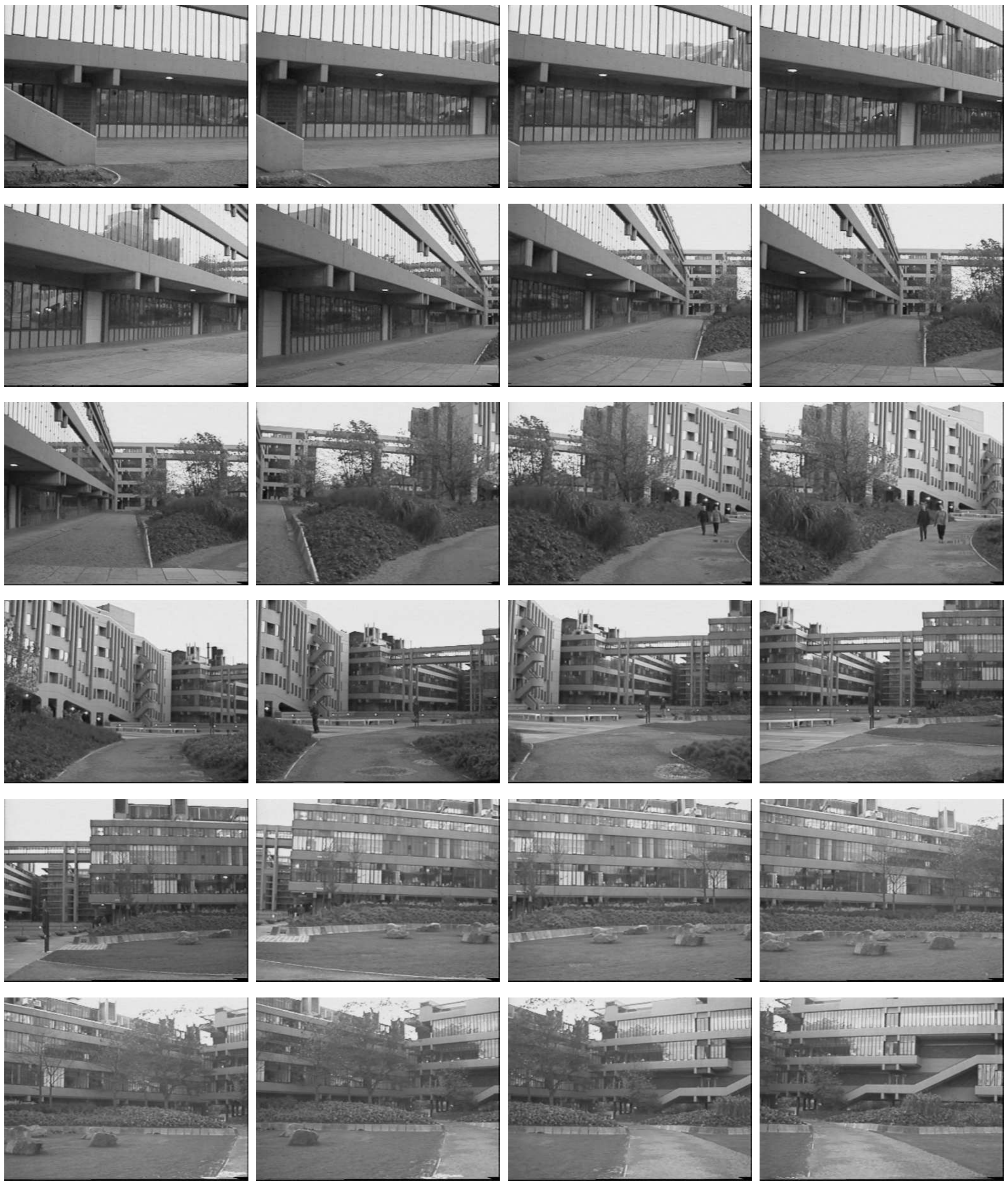


Figure C.3: Images 36 to 59

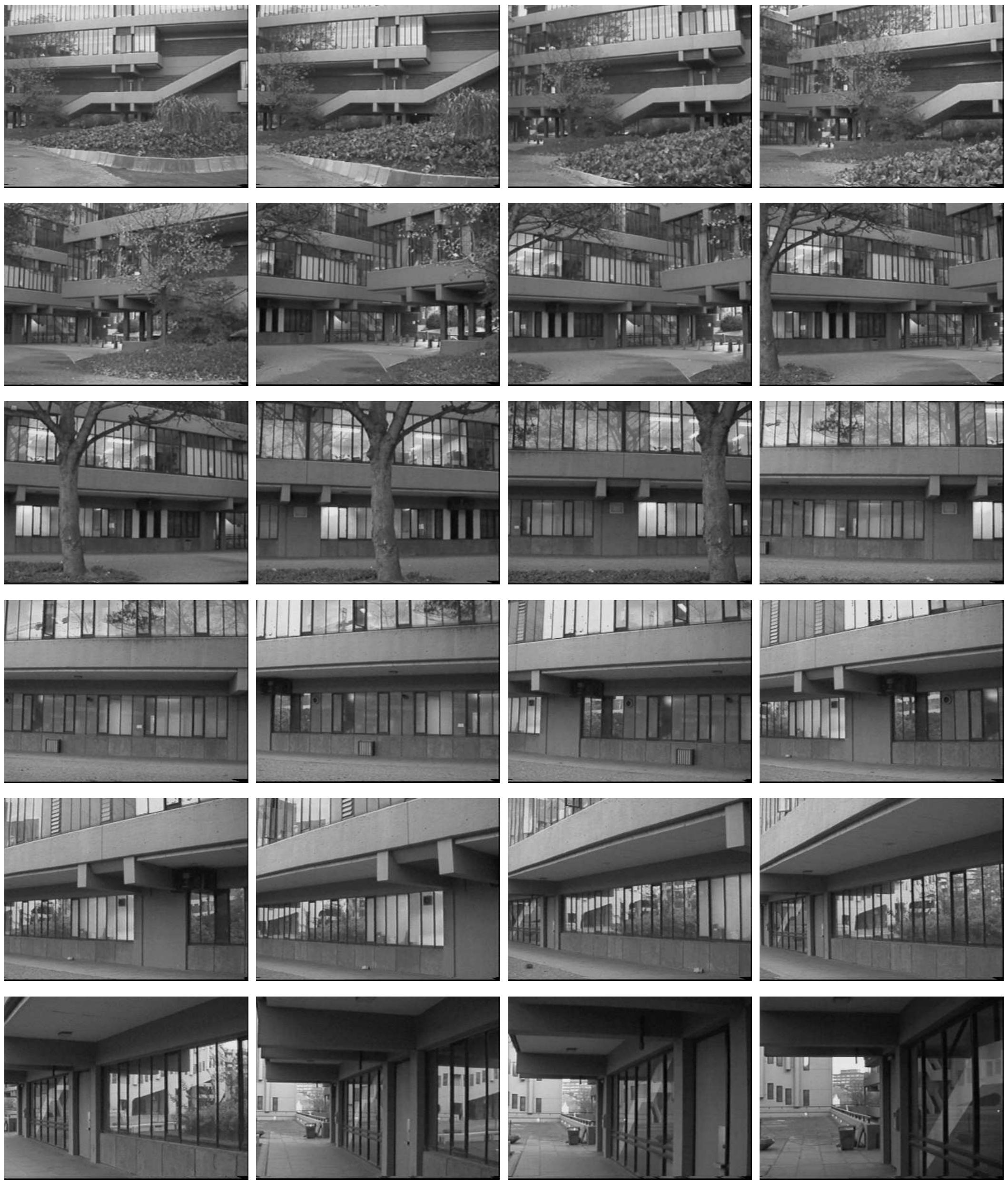


Figure C.4: Images 60 to 83

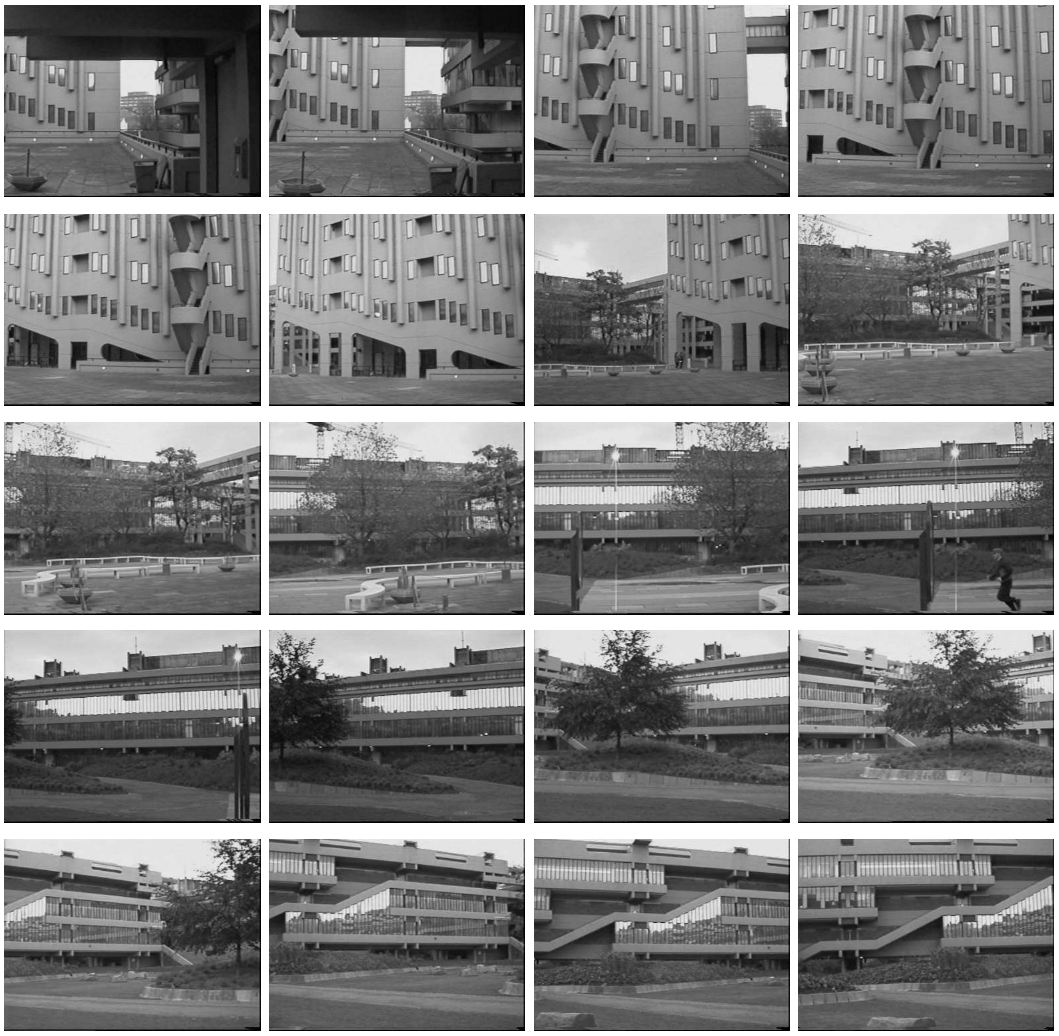


Figure C.5: Images 84 to 104

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. PAMI*, 7(4):383–401, 1985.

[2] M. Armstrong, A. Zisserman, and P. Beardsley. Euclidean structure from uncalibrated images. In Edwin Hancock, editor, *British Machine Vision Conference 1994*, volume 2, pages 509–518, University of York, September 1994. BMVA Press.

[3] D.H. Ballard. Generalising the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[4] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In B. Buxton and R. Cipolla, editors, *Computer Vision - ECCV96*, volume 2 of *Lecture Notes in Computer Science (1065)*, pages 683–695, Cambridge, UK, 1996. Springer-Verlag.

[5] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[6] J.R. Beveridge and E.M. Riseman. Optimal geometric model matching under full 3D perspective. *Computer Vision and Image Understanding*, 61(3):351–364, May 1995.

[7] T.O. Binford and T.S. Levitt. Quasi-invariants: Theory and exploitation. In *Proc. DARPA Image Understanding Workshop*, pages 819–829, 1993.

[8] B. Boufama and R. Mohr. Epipole and fundamental matrix estimation using virtual parallax. In *5th International Conference on Computer Vision*, pages 1030–1036, MIT, Cambridge, Massachusetts, June 1995. IEEE Comp. Soc. Press.

[9] T. Buchanan. Photogrammetry and projective geometry - an historical survey. In *Integrating Photogrammetric Techniques With Scene Analysis and Machine Vision*, volume 1944 of *SPIE*, pages 82–91. SPIE, 1993.

[10] R.T. Chen and C.R. Dyer. Model-based recognition in computer vision. *ACM Computing Surveys*, 18(1):67–108, March 1986.

[11] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, 1993.

[12] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Computer Vision - ECCV92*, volume 588 of *Lecture Notes in Computer Science*, pages 563–578, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[13] O.D. Faugeras, Q.-T. Luong, and S.J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Computer Vision - ECCV92*, volume 588 of *Lecture Notes in Computer Science*, pages 321–334, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[14] D. Fischer, R. Nussinov, and H.J. Wolfson. 3D substructure matching in protein molecules. In *Lecture Notes in Computer Science*, volume 644, pages 136–150. Springer-Verlag, 1992.

[15] 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.

[16] D.M. Gavrila and F. C.A. Groen. 3D object recognition from 2D images using geometric hashing. *Pattern Recognition Letters*, 13(4):263–278, April 1992.

[17] A.H. Gee and R. Cipolla. Fast visual tracking by temporal consensus. Technical Report CUED/F-INFENG/TR 207, University of Cambridge, Dept. of Engineering, February 1995.

[18] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1983.

[19] W.E.L. Grimson. *Object recognition by computer: the role of geometric constraints*. MIT Press, 1990.

[20] W.E.L. Grimson. On the sensitivity of geometric hashing. In *3rd International Conference on Computer Vision*, pages 334–338. Computer Society Press, 1990.

[21] W.E.L. Grimson and T. Lozano-Perez. Model-based recognition and localisation from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[22] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. of the Fourth Alvey Vision Conference (Manchester University, 31st August–2nd September)*, pages 147–152. The University of Sheffield Printing Unit, 1988.

[23] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. CVPR92*, pages 761–764, 1992.

[24] R.I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision - ECCV92*, pages 579–587. Springer-Verlag, 1992.

[25] R.I. Hartley. Euclidean reconstruction from uncalibrated views. In Joseph L. Mundy, Andrew Zisserman, and David Forsyth, editors, *Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*, pages 239–256. Springer-Velag, Second Joint European-US Workshop, Ponta Delgada, Azores, Portugal, October 1993.

[26] R.I. Hartley. In defence of the 8-point algorithm. In *5th International Conference on Computer Vision*, pages 1064–1070, MIT, Cambridge, Massachusetts, June 1995. IEEE Comp. Soc. Press.

[27] R.I. Hartley and J.L. Mundy. The relationship between photogrammetry and machine vision. In *Integrating Photogrammetric Techniques With Scene Analysis and Machine Vision*, volume 1944 of *SPIE*, pages 92–105. SPIE, 1993.

[28] A. Held. Piecewise shape reconstruction by incremental factorisation. In *British Machine Vision Conference 1996*, pages 333–342, 1996.

[29] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[30] P.V.C. Hough. Method and means for recognising complex patterns. US Patent 3,069,654, 1962.

[31] T.S. Huang and O. Faugeras. Some properties of the e matrix in two view motion estimation. In *IEEE Trans. Patt. Anal. Mach. Intel.*, volume 11, pages 1310–1312, 1989.

[32] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal of Computer Vision*, 5(2):195–212, 1990.

[33] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics and Image Processing*, (44):87–116, 1988.

[34] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Proc. CVPR88*, pages 335–344, 1988.

[35] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. On recognition of 3-d objects from 2-d images. In *Proc. ICRA*, pages 1407–1413, April 1988.

[36] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578–, October 1990.

[37] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *2nd International Conference on Computer Vision*, pages 238–249, Tampa, Fl. USA, September 1988. Computer Society Press.

[38] B. Lamiroy and P. Gros. Rapid object indexing and recognition using enhanced geometric hashing. In B. Buxton and R. Cipolla, editors, *Computer Vision - ECCV96*, volume 2 of *Lecture Notes in Computer Science (1065)*, pages 59–70, Cambridge, UK, 1996. Springer-Verlag.

[39] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[40] Q-T. Luong, R. Deriche, O. Faugeras, and T. Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. Technical Report 1894, INRIA, Sophia-Antipolis, April 1993.

[41] Q-T. Luong and O. Faugeras. The fundamental matrix: Theory, algorithms and stability analysis. *International Journal of Computer Vision*, (17):43–75, 1996.

[42] S.J. Maybank. Properties of essential matrices. *International Journal of Imaging Systems and Technology*, 2:380–384, 1990.

[43] P.F. McLauchlan and D.W. Murray. A unifying framework for structure and motion recovery from image sequences. In *5th International Conference on Computer Vision*, pages 314–322, MIT, Cambridge, Massachusetts, June 1995. IEEE Comp. Soc. Press.

[44] P.M. McLauchlan and D.W. Murray. A unifying framework for structure and motion recovery from image sequences. Technical report, Robotics Research Group, Dept. of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK, 1994.

[45] R. Mohan, D. Weinshall, and R.R. Sarukkai. 3D object recognition by indexing structural invariants from multiple views. In *4th International Conference on Computer Vision*, pages 264–268, May 1993.

[46] T. Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. Technical Report CMU-CS-94-158, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, June 1994.

[47] J.L. Mundy and A.P. Zisserman. *Geometric invariance in computer vision*. MIT Press, 1992.

[48] C.J. Poelman. *The Paraperspective and Projective Factorization Methods for Recovering Shape and Motion*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1995. CMU-CS-95-173.

[49] C.J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In Jan-Olof Eklundh, editor, *Computer Vision - ECCV94*, volume 801 of *Lecture Notes in Computer Science*, pages 97–108, Stockholm, 1994. Springer-Verlag.

[50] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1986.

[51] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Tippett et al., editor, *Optical and electro-optical information processing*, pages 159–197. The MIT Press, 1965.

[52] C. Rothwell, G. Csurka, and O. Faugeras. A comparison of projective reconstruction methods for pairs of views. In *5th International Conference on Computer Vision*, pages 932–937, MIT, Cambridge, Massachusetts, June 1995. IEEE Comp. Soc. Press.

[53] C. Rothwell, G. Csurka, and O. Faugeras. A comparison of projective reconstruction methods for pairs of views. Technical Report 2538, INRIA, April 1995.

[54] C.A. Rothwell, A. Zisserman, D.A. Forsyth, and J.L. Mundy. Using projective invariants for constant time library indexing on model based vision. In *British Machine Vision Conference 1991*, pages 62–70, 1991.

[55] C.A. Rothwell, A. Zisserman, D.A. Forsyth, and J.L. Mundy. Planar object recognition using projective shape representation. *International Journal of Computer Vision*, (16):57–99, 1995.

[56] C.A. Rothwell, A. Zisserman, J.L. Mundy, and D.A. Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proc. CVPR92*, pages 109–114, 1992.

[57] S.M. Smith. A new class of corner finder. In *British Machine Vision Conference 1992*, pages 139–148, 1992.

[58] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - full report on the orthographic case. Technical Report CMU-CS-92-104, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, March 1991.

[59] P.H.S. Torr. Oxford: Vanguard project home page. http://www.robots.ox.ac.uk:5000/ vanguard/index.html.

[60] P.H.S. Torr and D.W. Murray. A review of robust methods to estimate the fundamental matrix. Technical report, Robotics Research Group, Department of Engineering Sceience, Oxford University, 1996.

[61] P.H.S. Torr and A. Zisserman. Robust parameterisation and computation of the trifocal tensor. In *British Machine Vision Conference 1996*, pages 655–665, 1996.

[62] P.H.S. Torr, A. Zisserman, and S.J. Maybank. Robust detection of degenerate configurations for the fundamental matrix. In *5th International Conference on Computer Vision*, pages 1037–1042, MIT, Cambridge, Massachusetts, June 1995. IEEE Comp. Soc. Press.

[63] F.C.D. Tsai. Geometric hashing with line features. *Pattern Recognition*, 27(3):377–389, 1994.

[64] R.Y. Tsai. Synopsis of recent progress on camera calibration for 3D machine vision. In *The Robotics Review*. MIT Press, 1989.

[65] R.Y. Tsai and T.S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. In *IEEE Trans. Patt. Anal. Mach. Intel.*, volume 6, pages 13–27, 1984.

[66] S. Ullman. *The interpretation of visual motion*. The MIT Press, 1979.

[67] H. Wang and M. Brady. Corner detection for 3D vision using array processors. In *Proc. BARNAIMAGE-91*, Barcelona, 1991. Springer-Verlag.

[68] D. Weinshall and C. Tomasi. Linear and incremental acquisition of invariant shape model from image sequences. In *4th International Conference on Computer Vision*, pages 675–682, May 1993.

[69] C. Wiles and M. Brady. On the appropriateness of camera models. In B. Buxton and R. Cipolla, editors, *Computer Vision - ECCV96*, volume 2 of *Lecture Notes in Computer Science (1065)*, pages 228–237, Cambridge, UK, 1996. Springer-Verlag.

[70] H.J. Wolfson. Model-based object recognition by geometric hashing. In O. Faugeras, editor, *Computer Vision - ECCV90*, Lecture Notes in Computer Science (427), pages 526–536, Antibes, France, 1990. Springer-Verlag.

[71] Z. Zhang, R. Deriche, O. Faugeras, and Q-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Technical Report 2273, INRIA, May 1994.