Periscopic Stereo and Large-Scale Scene Reconstruction

C. E. Moxey

A thesis submitted for the degree of

Doctor of Philosophy.

Department of Electronic Systems Engineering

University of Essex

July 2002

Abstract

The capture of three dimensional structure from two dimensional images has received considerable attention in computer vision. Existing work has concentrated on use of stereo camera systems and the reconstruction of small objects. Recently, single cameras in motion have been used to capture sections of scenery which are subsequently reconstructed by skilled technicians with a selection of computer vision and graphical modelling tools. However, large-scale, automated, reconstruction of scenery is limited by the “where to look next” problem. A number of imaging systems have been proposed to solve this problem but none have been realized. *Periscopic Stereo* is a novel concept which implements stereo imaging using a single camera. A rotating mirror scans the horizon while a fixed relative geometry is maintained between the virtual stereo cameras.

This dissertation presents, for the first time, a practical design for a periscopic stereo head and investigates the computer vision tools necessary for 3D reconstruction from periscopic image data. It identifies two possibilities for processing periscopic image data. “Corrected”, where a two dimensional rotation is applied to the image plane prior to standard stereo processing, or, “uncorrected” which ignores the “tumbling” effect inherent in periscopic image data until the final stage of reconstruction, where the “late” correction circumvents the problem, apparent in many existing stereo algorithms, of resolving disparity measurement in imaged scene structure which is parallel with corresponding epipolar lines.

Many of the existing stereo processing tools used in the course of this research require little modification, but have all revealed issues requiring resolution not immediately apparent in previous treatments. This investigation stops short of the actual construction of 3D models but presents a method of generating the sets of depth data required for large-scale scene reconstruction. Feature extraction, image data correspondence, camera calibration and the generation of depth information from periscopic image data are all covered in the context of this dissertation. In particular a new method of combining existing camera calibration techniques, termed “calibration in a box”, is presented together with conclusions regarding the tools and techniques employed.

While periscopic stereo is still in development, it is the only imaging system, reported to date, which is likely to be capable of large-scale, autonomous, 3D scene reconstruction, with particular application to remote operation in hazardous environments.

Contents

List of Figures

List of Tables

Symbols and Glossary

$\boldsymbol{x}$ - 2D image vector in normal Euclidean coordinates.

$\boldsymbol{X}$ - General 3D vector in normal Euclidean coordinates.

$\tilde{\boldsymbol{x}}$ - 2D vector in Projective, or Homogeneous, coordinates.

$\tilde{\boldsymbol{X}}$ - 3D vector in Projective, or Homogeneous, coordinates.

X_w - Euclidean world coordinate.

X_c - Euclidean camera coordinate.

X_i - Euclidean image coordinate.

$\mathcal{P}^n$ - General Projective space.

$\mathcal{R}^n$ - General Euclidean space.

$\mathcal{R}^2$ - Euclidean plane.

$\mathcal{R}^3$ - Euclidean space.

I_b - Interocular baseline distance between the two cameras in a stereo imaging system.

E - Essential matrix.

F - Fundamental matrix.

K - Camera matrix.

M - normalized Projection matrix.

P - full Projection matrix.

R - 3D Rotational matrix.

dpm - Degrees per minute.

dps - Degrees per second.

fps - Frames per second.

rpm - Revolutions per minute.

AR - Augmented Reality.

CofG - Centre of Gravity, given by the ‘first moment’ of a region.

NMS - Non-Maximum Suppression, method to determining the single, most accurate element for an edge response.

SUSAN - Smallest Univalue Segment Assimilating Nucleus.

VE - Virtual Environment.

VR - Virtual Reality.

Affine transformation - a transformation that preserves parallel lines. Algebraically this is any invertible linear transformation.

Canonical configuration - stereo system with coplanar image planes where corresponding raster lines are collinear.

Collinear - lie on the same line.

Collineation - linear transformation in projective space, also referred to as an *homography*.

Coplanar - lie on the same plane.

Correspondence - attempting to match two image points that correspond to the same point in the scene.

Disparity - the horizontal and/or vertical displacement of corresponding elements.

Edgels - short hand terminology for Edge Elements.

Epipolar geometry - the geometry between two cameras that image the same scene.

Essential Matrix - mathematical representation of the epipolar geometry of two calibrated cameras viewing the same scene.

Euclid - Greek mathematician c. 300 B. C., author of the *Elements*, which introduced many of the foundations of geometry, and of the *Optics*, which considers the projection of rays of light and the seeds for the idea of the *vanishing point*.

Euclidean geometry - ordinary, flat or real space, geometry with all measures of curvature equal to zero.

Focal length - distance from the centre of projection to the image plane.

Fronto-parallel - a plane, or scene, in front of the image plane which is coplanar and has the same alignment or frame of reference.

Fundamental Matrix - mathematical representation of the epipolar geometry of two uncalibrated cameras viewing the same scene.

Homogeneous Coordinates - system of coordinates with $(n + 1)$ components in which any scalar multiple of a point, or a line, is the same point, or line.

Homogeneous Region - contains the same elements - is uniform.

Isotropic filter - has an equal response in all directions.

Near real-time - action from a system within a perceived acceptable limit, but not necessarily within a fixed time frame specified by some physical, real-world requirement.

Outliers - data that are in gross disagreement with the postulated model.

Photogrammetry - the use of photographs, or imaginary, for the measurement of distances or dimensions.

Perspective projection - sometimes called *central projection*, spatial transformation of 3D to 2D performed by an imaging device.

Rectification - process of re-sampling the image data from a stereo system in order to return the camera geometry to its canonical configuration.

Silhouette frame - the frame induced into the image data by a rotational correction about the optical axis.

Spline - edge string that is modelled by a piece-wise polynomial.

Stereopsis - two view, or stereo, imaging.

Vanishing point - point at projective infinity, to which all parallel 3D lines viewed in the image plane converge.

Acknowledgments

I would like to thank my supervisor, Peter Noakes, the head of the Virtual Applications, Systems and Environments Laboratory, Dr Adrian Clark and my fellow postgraduate students; Neill Newman, David Johnston and Mike Lincoln, and all the support staff of the Department of Electronic Systems Engineering at The University of Essex.

I would especially like to thank, Dr Steve Sangwine, for his recent support and for giving me the opportunity to continue with image processing and computer vision research in academia. Finally, I would like to thank my girlfriend, Chris Ward, for her patience, understanding and continued support over the past few years.

Notes

1. This report has been compiled using the 'XEmacs' text editor under Red Hat Linux ver. 7. 1 and typeset with LaTeX 2_ε and $\mathcal{AMS}$-LaTeX.
2. All the sketches were drawn using 'Xfig' and subsequently included in EPS format. Some diagrams include embedded LaTeXmath symbols via PSTEX format using 'fig2dev'.
3. All images are were originally created in TIFF format, converted into PNG format using the 'Xv' imaging display and editing package and subsequently converted to EPS using 'convert' available for Linux from ImageMagick (`http://www.imagemagick.org/`).
4. The references were created using BibTeX with `alpha.bst` bibliography style.

Chapter 1

Introduction

Considerable attention has been given, in recent years, to improving the techniques required to recover sufficiently accurate three dimensional data from two-dimensional images, in order to reconstruct a model of an imaged scene, or recognize objects within it. Finding solutions to problems of this kind has been a central goal in computer vision for many years. However, the various image processing and data analysis techniques required to realize functional systems are only now maturing. The complexity of elements such as calibration, feature extraction, correspondence and registration will continue to ensure that the design and development of robust systems capable of delivering the desired results will remain a considerable challenge. The aims of projects in this area of computer vision research, often referred to as *three dimensional* (3D) computer vision, have been varied but generally fall into two main groups, reconstruction and recognition. Historically they were considered the same. The research presented in this dissertation deals exclusively with techniques related to scene reconstruction and no attempt is made to deal with any of the issues concerning the recognition of specific objects.

The number of applications for 3D reconstruction from imagery have been steadily increasing and Table 1.1 gives a broad overview of the main areas with a few examples which are used to introduce or support concepts discussed herein. The type of system and the level of constraints which can be applied to simplify the problem are given together with type of reconstructed model and its accuracy.

Application Area	*Type of System and Level of Constraint*	*Model Requirement*	*Model Accuracy*	*Example Applications and Projects*
industrial inspection	laser stripping or precision stereo cameras, highly constrained environment	initially none, more recently merged with CAD tools	none or high	numerous[1][2][3]
aerial or satellite survey	precision stereo cameras, few constraints used	initially none, more recently surface 2.5D models	none or medium to high	crop yield, terrain modeling[4]
medical imaging	laser/MRI scanning, stereo cameras highly constrained environment	surface 2.5D of deformable objects (tissue) and full 3D models (tumors, bones)	very high	clinical diagnosis, surgical planning, such as 'Visible Human'[5]
virtual reality	hand-held video camera minimal constraints	photo-realistic surface models (scenery) some 3D models (objects in local vicinity)	low since appearance is more important	virtual tour guides, interior design, such as 'VANGUARD'[6]
augmented reality	stereo or single cameras with markers, known motion TV video cameras, highly constrained	photo-realistic full 3D models	very high due to the need for registration of the model with the real world	telepresence tour guides 'Virtual Studio'[7]
mobile robotics	laser stripping, stereo cameras, alternative stereo imaging, some constraint possible	surface 2.5D (scenery) and full 3D models (local vicinity)	medium to high depending on application	robot navigation and mission planning, such as 'Pioneer'[8], telepresence with environment interaction such as 'NARVAL'[9]

Table 1.1: Application areas for 3D reconstruction from imagery.

Table 1.1 is intended as a guide, not a definitive breakdown for areas of application.

[1]http://www.ipb.uni-bonn.de/ipb/projects/projects.html

[2]http://www.ndt.net/article/v05n05/saxena/saxena.htm

[3]http://www.terarecon.com/recon_ind.shtml

[4]http://www.aca-net.com/

[5]http://www.crd.ge.com/esl/cgsp/projects/medical/

[6]http://www.robots.ox.ac.uk/~vanguard/

[7]http://www.bbc.co.uk/rd/projects/virtual/

[8]http://robotics.jpl.nasa.gov/tasks/pioneer/homepage.html

[9]http://www.isr.ist.utl.pt/vislab/projects.html

Initially, most reconstruction projects were concerned with the inspection of manufactured parts or objects on the ground, imaged by aerial photography or satellites. In general the aim was just the recovery of accurate measurement of these objects. The use of imagery, traditionally photographs, for the precise measurement of distances or dimensions is known as *photogrammetry*. This definition, together with many others that appear herein, is given in the glossary in the preliminary section of this dissertation. More recently the reconstruction techniques have merged with Computer Aided Design (CAD) tools for re-engineering applications [IH98, Pra00].

The medical profession has also made effective use of reconstruction techniques for diagnosis and surgical planning. In such applications the requirement is to produce highly accurate models with shortest possible delay. This has generally involved highly constrained environments, applying scan imaging techniques, and relatively small (in the physical volume sense) models.

The latest area of applications has been interested in the recovery of structure to create realistic 3D models for use in Augmented Reality (AR) or Virtual Reality (VR) systems, such as those found in architectural planning and interior design. There have been considerable advances made in the appearance and level of accuracy of the models for these applications. However, as with the previous examples, the processing of the imagery and the reconstruction the 3D model data is carried out "off-line", prior to the intended use.

There are a number of examples with successful implementation for the applications in the first three rows of Table 1.1, some more constrained than others. However, there are only a few operational examples of the applications in the lower three rows. The reasons for this will be explained in Chapter 2. Periscopic stereo is presented in this dissertation as a possible solution for applications in these last three areas. As will be demonstrated throughout this dissertation its unique imaging geometry offers many advantages which simplify the processing of the image data and the recovery of the depth and structure required to reconstruct a model of the scene. However, periscopic stereo is not expected to be able to address applications in the first three areas identified in Table 1.1 since its unique geometry also introduces limitations which makes it unsuitable for those areas. Again, this will become apparent later.

The motivation for the research presented in this dissertation stems from a desire to be able to interact with a virtual representation of the real world, where direct interaction is either impossible or impractical. The intention is therefore to extend the boundaries of existing systems by exploring the possibility of producing 3D models of the imaged scene "online", in *near* real-time. Applications for this project could be found in both VR and AR areas but are more in tune with the requirements of remote exploration where the inspection and mapping of hazardous environments, such as collapsed buildings, mines or deep water trenches and caves, is conducted by mobile robots, often in a fully autonomous mode. An example of this is 'NARVAL' which is an Esprit–LTR project studying the performance and use of 'fully' autonomous mobile robots [RRT01].

Apart from the external benefits gained from such data, improving robotic perception by providing a form of "visual map" of the robot's environment would greatly aid robotic task planning and mission analysis. The concept of a visual map also provides a human "view-able" record of the robot's iteration within the environment and thus aids the design and development of robust, fault tolerant systems.

The concept of large-scale models of the real world raises questions concerning the scale and efficiency of reconstruction. This is directly relevant to robotic applications and remote exploration. The definition of small- and large-scale models could be open to debate, so for the purpose of the ideas and discussion presented in this dissertation the following distinction is made.

> Small-scale reconstruction applies to specific objects or parts of a scene where the imaging system pans around the periphery. This is the imaging system is imagined to be on the outside looking in.

Large-scale reconstruction applies to large areas of the surrounding scenery where the imaging system can be imagined as being on the inside looking out.

In some applications this distinction is obvious and in others it is more subtle. The first three application areas in Table 1.1 can be categorized as small-scale, even though the models are often large and complex. Conversely the lower three application areas can be regarded, in general, as large-scale. However, few of the examples in these areas could claim to perform successful large-scale reconstruction of the surrounding scene. The reasons for this will be explained in detail in Chapter 2. However, the next few paragraphs will briefly expand on this.

The more information that can be derived about the robot's surrounding environment the greater its ability to operate effectively within it. However, the generation of large amounts of data about that environment has distinct implications for processing efficiency and data storage. Apart from robotic systems, the problem of the scale is found in other applications. Recent advances in VR techniques have allowed the television and film industry to create 'Virtual Studio's [TJNU97] where actors are filmed performing in a large empty studio and the background scenery is added afterward during editing and post production. Unlike the traditional *croma-key* technique of replacing the blue background of the real studio with some other scene, where the camera must remain fixed, the virtual studio system allows for free movement of the camera and interaction between the actor and the virtual environment. This is only possible because the position of the camera and the actors within a full 3D VR model of the film set, or scenery, are known at all times. This technique requires detailed, *large-scale*, scenery models which are constructed manually over many hours by skilled technicians using CAD modeling or computer graphics tools. The concept of a mobile robotic system which could survey the local environment in order to reconstruct an accurate, photo-realistic, 3D model of the scene, or film set, would probably be an attractive alternative. As yet, there is no evidence of a system capable of producing such large-scale 3D models of the imaged environment autonomously.

In all of the applications mentioned here, fast access to the reconstructed model has often been desirable but has rarely been possible. Many of the systems that have been developed to date have produced impressive results. However, in general, these have involved the application of strict constraints and been computationally intensive. These constraints range from static scenes with fixed illumination to known motion and/or fixed camera geometry. While the application of constraints in this area of research is almost mandatory their use should be minimal and always viewed in the context of the target application. Moreover, in the case of applications, such as robotic exploration, the requirement for large-scale models of the surrounding environment suggests the need for a system which can deal with totally unconstrained, dynamic scenes.

The two historic methods of image data acquisition for reconstruction have been laser scanning and stereo imaging. Both these have been successful since they provide the highest possible accuracy of any system to date. However, they have been accompanied by their own set of limitations. Laser scanning systems are expensive, require calibration and a degree of expert use in order to gain good overall coverage. As such, they are more suitable for scanning static objects than large-scale scenes. Active vision systems, employing an actuated stereo head, introduce known camera geometry which simplifies many of the problems. These system are much better for reconstructing scenes but still suffer from the problem of "where to look next"; that is how to synchronize the camera motion for efficient capture of the scene for reconstruction. Hence such these systems require complex control algorithms and/or give poor coverage.

Recent advances in camera calibration [Har94] and the analysis of *homography* [PZ98, ST98] have allowed for the relaxation of many of the constraints and a solution to the inherent cyclic problem of feature correspondence versus camera calibration; the solution of one greatly aids the other. This has lead to the introduction of systems which employ only a single, hand-held, video camera. These systems offer a major break through, since they allow for arbitrary motion of the camera and avoid the need for the initial calibration of the camera using precision calibration objects which are placed in the scene. However, even a simple, hand-held video camera requires some expertise in order to capture the whole scene. There is also the question of the considerable amount of image data produced by a video camera and how much of that data is useful for reconstruction. If the camera motion between subsequent frames is small then the stereo effect is minimal and the frames can not be used for scene reconstruction. This suggests a large amount of redundant image data will always be present in such systems. Furthermore, autonomous reconstruction from such image sequences would require some measure of minimum disparity, which, without known motion of the camera, is scene dependent. Consequently, robotic applications for such systems are unlikely.

The introduction of the single camera reconstruction systems has given rise to the concept of the "uncalibrated" approach and is often thought to be in competition with the traditional calibrated approach. This is extremely misleading since the approach does include a method of calibration, but without the need for an object in the scene. It also has an inherent limitation in that it can only recover accurate scale of measurement, not absolute measurement. This approach can therefore only be used to reconstruct, at best, a scaled model of the scene. The reason for this will be covered later in Chapters 2 and 6. A limited but more immediate support for the above statements can be gained for reference to Table 2.1. Reconstructions from single, hand-held, cameras in free motion are therefore only effective for VR applications where the accuracy of measurement is less important than the visual appearance or the ease with which the scene can be captured. For most other applications, the necessity for accurate measurement requires the "calibrated" approach. Again this is explained in considerably more detail in Chapters 2 and 6 with numerous supporting references. The terms "calibrated" and "uncalibrated approaches" will not be used in the rest of this dissertation for the reasons stated here.

The obvious alternative to these imaging systems, that solves some of the conflicting issues, has been the idea of simulating the stereo effect with single camera that *mechanically* scans the horizon. These will be discussed in detail later in Chapters 2 and 3. Several such systems have been proposed but all of them have relied on either complicated mechanics, specialized mirrors and prisms or employed proprietary software and none, to date, have successfully demonstrated 3D reconstruction on either scale. However, the imaging system, termed as *Periscopic Stereo* by its inventors, has a unique imaging geometry which allows for the use of many of the latest methods applicable to standard stereo imaging. This is achieved by creating multiple instances of a conventional stereo vision system by using a single camera with a rotating, flat, mirror and processing sets of image sequences to recover the true, Euclidean models of large-scale scenes via incremental estimation of the observed 3D geometry. Although the imaging geometry of periscopic stereo has a number of advantages its “cross-eyed” configuration means that it can not be used with *affine* camera models to recover structure which has a small depth compared to its size. That is periscopic stereo is effectively long sighted and can not be used in applications requiring high accuracy of small structure.

The research presented in this dissertation draws together a number of algorithms and techniques reported in the literature on 3D reconstruction and applies, suitably modified versions, to the image data captured by a periscopic stereo system. While these modifications, taken as a whole, could be regarded as the development of proprietary software, the author would argue that individually their origin in standard, proven, stereo techniques does not justify the label of “specialized software”.

1.1 Research Objectives and Structure of this Thesis

The aims of this research are twofold. The first is to demonstrate that periscopic stereo is more than just a concept but is a realizable imaging system. In order to achieve this each of the individual processes in the system that implements 3D reconstruction are reviewed. In each case the image processing and computer vision techniques normally used in stereo imaging systems are assessed for their compatibility with the peculiarities of periscopic stereo image data. The second aim of this research is to demonstrate that periscopic stereo has many inherent advantages which make it uniquely capable of producing large-scale scene reconstructions.

This thesis begins with a short review of 3D reconstruction from imagery, identifying some of the recent influential research in this area and sources of useful information. Specific advances in individual topics will be identified in the relevant chapters, each of which will begin with its own brief introduction. A comment on the choice of software framework for research into image processing and computer vision techniques is included since many of those reviewed are from established methods contained therein.

Chapter 3 reviews the concept of *periscopic stereo* and reworks the original analysis to yield a practically realizable imaging system. The design and construction of a periscopic stereo head are presented together with advice on operational requirements and use. The inherent nature of rotating image data is recognized and the two possible methods of correction are identified. These competing methods become a continuing thread throughout this dissertation. To date a functional system which produces complete large-scale scene reconstruction is still in development. Consequently, many advantages expected from periscopic are yet to be realized. All the experimentation presented in this dissertation has been conducted on image data captured from a "simulated" periscopic stereo head using a suspended camera and a mirror on a turn table. A description of this system and the capture of the image data is included.

3D reconstruction relies on accurate, sub-pixel, localization of image features. Established, derivative-based, methods of feature extraction suffer from an inherent weakness which has particular implications for reconstruction. That being the inherent use of filtering in derivative based methods induces localization errors in the derived features. While this is arguably small, any induced error in the initial stages of processing will undoubtedly be propagated and should therefore be avoided where possible. Chapter 4 briefly reviews the requirements of the feature extraction process for reconstruction, together with the SUSAN algorithm [SB97] which is reportedly designed to address the limitations of standard methods. Modifications to the implementation and operation of both one- and two-dimensional, SUSAN, feature detectors are included.

Matching image features that correspond to the same feature in the world is essential for the recovery of depth information. This has been a topic of considerable interest to the computer vision research community. Chapter 5 reviews the standard approach to the correspondence problem, identifying the constraints that are often employed in the solution. An algorithm, applicable to a stereo imaging system with small, known, relative motion between the views, is presented. This algorithm, based on existing techniques, incorporates a practical compromise which simulates *shear* correlation [LTM94] with image patches warped according to the specific relative camera geometry imposed by periscopic stereo.

All practical 3D reconstruction systems require some form of camera calibration. The ability to calibrate an imaging system and to maintain some measure of its continued accuracy is fundamental to its validation and acceptance. The research presented in this dissertation therefore pays particular attention to the calibration of periscopic stereo, specifically for the purpose of large-scale reconstruction. Chapter 6 reviews the most widely referenced techniques for camera calibration and introduces a new technique, essentially applicable to periscopic stereo but valid for standard stereo camera systems. This method, termed "*Calibration in a Box*", combines the epipolar calibration of a stereo system with standard grid calibration in a novel method that accommodates autonomous calibration prior to, and re-calibration during, operational use. Chapters 4, 5 and 6 contain the results of a number of experiments with the individual techniques.

Chapter 7 completes the process of generating 3D structure from images with a brief review of reconstruction of multiple depth images. The generation of large-scale models of the scene is discussed and the advantages of delaying the correction for the rotating image data, inherent in periscopic stereo, is presented. Chapter 8 summarizes the major points presented in this dissertation and concludes that periscopic stereo is a viable imaging system for large-scale 3D scene reconstruction. A discussion of the future direction of research into large-scale reconstruction is also given together with recommendations for the implementation of the reviewed software techniques.

Chapter 2

Review of Relevant Work and Background Information

A person's ability to estimate depth stems from the simple fact that they have two eyes. However, humans also use secondary cues from the imaged scene to estimate structure. Artists have used depth in pictures to convey the concept of structure in two dimensional (2D) representations of three dimensional (3D) objects for centuries. Getting a machine to estimate 3D information from 2D images in order to discover something about the structure of the imaged world, is not exactly a new concept either. Marr [Mar82] proposed various ideas about 3D, scene reconstruction in the late 1970's and early 1980's which are still referenced today. A paper by Tenenbaum proposing scene modelling using various "shape from X" techniques appeared in *Image Modeling*, edited by Rosenfeld in 1980 [TFB80]. The background to research in this area is therefore considerable. To carry out an exhaustive review of all the work in this area would be excessively time consuming and not particularly productive, since much has been superseded by later work. Most of the following review will therefore be constrained to the last decade or so. However, a good collection of general papers covering the major achievements in computer vision between the late 1970s and mid 1980s can be found in *Readings in Computer Vision* [FF87]. Included there is the noted paper by Brooks on *Visual Map Making for a Mobile Robot* which has been an inspiration of many projects, including this one.

This chapter begins with a brief overview of 3D scene reconstruction and continues with the basic concepts that are encountered within this dissertation. Some of the influential research connected with 3D reconstruction projects in recent years are also identified. An introduction to 3D reconstruction and the associated computer vision techniques can be obtained from a number of comprehensive text books on computer, and/or machine, vision [Dav97, JKS95, SHB99]. More specialized texts which give an excellent review of 3D Computer Vision are by Klette, Schlüns and Korschan [KSK98] and Faugeras [Fau93]. Another excellent text, by Hartley and Zisserman [HZ00], has just been published. The book includes an extensive review of many techniques, particularly from the viewpoint of imaging geometry, and extends their earlier work [MZ92] which provided a thorough examination of the classic mathematical treatment of projective geometry by Semple and Kneebone [SK52]. In addition to these publications, an excellent resource for computer vision research is the collection of on-line tutorials and references maintained by the Computer Vision Laboratory of the Department of Artificial Intelligence at Edinburgh University. [1]

2.1 Overview of Scene Reconstruction

The process of image formation in a camera, as in the human eye, can be explained with the use of *pinhole* camera model. Rays of light reflected from the scene pass though a single point, called the *centre of projection*, and project on to a flat plane some distance behind. This spatial transformation of light from 3D objects onto a 2D image plane is known as *perspective*, or *central*, projection. This

[1]The CVonline website can be found at: `http://www.dai.ed.ac.uk/CVonline/`

geometrical relationship between the camera and objects in the scene can be modelled in terms of their respective *coordinate* systems. The basic theory of perspective projection via the transformation across these coordinate frames is provided in Appendix A.

In reality, the image on the plane at the rear of a camera is inverted. However, by moving the centre of projection back behind the image plane a geometrically equivalent model is produced without the inversion. In this configuration the various concepts of projection are easier to visualize. This model will, therefore, be used throughout this dissertation. Although some estimation of depth is possible with a single image, it requires a number of related cues within the image data and is often, only a rough approximation. The obvious method of recovering depth is to employ two views of the same scene and estimate depth from the disparity between corresponding features in the two images. The geometry of two cameras, referred to as *epipolar* geometry, is also covered in Appendix A.

From an overall "systems" viewpoint 3D reconstruction can be divided into three distinct processing stages;

1. Image Pre-processing.
2. Shape from *X*.
3. 3D model construction.

This is a bottom-up approach to reconstruction was proposed by Marr [Mar82] in the early 1980's and assumes little or no prior knowledge is given. This is still the basis of many systems including periscopic stereo. The alternative top-down approach assumes a priori knowledge of the scene or the objects within it and attempts to recognize elements based on models. This model-based approach has considerable merit in applications where the recognition of specific objects is paramount and there are a number of practical examples [Goa86, NFJ93]. However, the difficulty in defining models for arbitrary scenes or objects tends to make this approach less attractive for 3D reconstruction [SHB99, chap.9].

In the bottom-up approach the pre-processing stage involves the detection of basic features in the image such as points, lines and curves in order to estimate some useful geometrical cues about the structure. Once these features, or *image tokens*, have been identified they can be processed by one of a number of techniques that estimate some measure of the *shape* and/or depth of the objects in the scene. Using this "shape" terminology, the concept of using two corresponding views could be described as "shape from stereo". However, this is almost always referred to, simply, as *stereo matching* in binocular vision systems. Unfortunately, recovering depth for stereo is not as simple as it sounds. There are two inherent problems, explicitly related to each other, which must be solved in order to extract accurate estimates of depth. The first is to identify and match the *corresponding* features in the left and right images. The second is the calibration of the camera system itself. This involves estimating the internal camera parameters, which define the image formation, and the spatial relationship between the two camera positions. Both of these continue to be the subject of major research effort.

The remaining, "true" *shape* techniques calculate depth indirectly by first estimating the surface orientation from the image data and then integrating over a local area. These techniques have been grouped into:

- Shape from Motion (SFM), similar to stereo but with either a single camera or the scene in motion (usually the camera). This is the most widely used method.

- Shape from Shading (SFS), exploits the change in image intensity between corresponding image points of objects with known reflectance properties. This relies on fixed illumination and accurate reflectance properties of the objects within the scene, both of which are not usually known.

- Shape from Texture (SFT), uses the change in either the size or density of texture elements to determine orientation, either directly or by first determining the ‘vanishing point’. This is only useful if there are sufficient areas of regular texture.

- Shape from Focus (SFF), relies on the fact that only objects at a certain distance from the camera, depending on the ‘depth of field’, will be in focus. All other points will be blurred in proportion to the distance from that point.

- Photometric stereo, is similar to SFS but uses images with different scene illumination, captured by a static camera. Relies on knowing the surface reflectance of all objects and requires a static scene, so has limited use.

All these methods rely on knowing the depth of at least one point on any object in order to recover metric structure. Apart from SFM, these techniques are generally used to yield secondary cues about the structure of the scene and are often combined with the more direct, stereo based algorithms. The use of shape techniques, is beyond the scope of the research presented in this dissertation.

Having obtained depth estimates and/or some basic information about the geometry of the scene, a *disparity image* (often referred to as a range image, 2.5D sketch, or depth map) can be produced in order to display the inferred structure. At this stage some form of texturing could be used to improve visualization of the recovered scene. Such representation is still *view-centered* and not *object-centered*, which is required for fully interactive 3D representations, such as those used in most 3D CAD modelling tools. The final, 3D modelling stage therefore attempts to determine such object-centered descriptions from the basic depth information and secondary cues by employing either *model-based recognition* schemes or *imaged-based reconstruction* algorithms. The representations of 3D objects are largely dependent on which of these methods is chosen and there are now a number of possible alternatives which, themselves, fall into two distinct classes, *volumetric* and *surface* models. This last stage of the reconstruction process is often considered to be outside the remit of Computer Vision and relies heavily on techniques developed for Computer Graphics. An excellent, well-referenced, text covering this discipline is by Foley, van Dam, Feiner and Hughes[FvDFH97]. Again, the subject of modelling 3D data is beyond the scope of the research presented here.

This provides a cursory overview and does not consider the “type” of 3D reconstruction required or the precise nature of image data captured from the scene. Both of these facts are closely related and fundamental to any further consideration.

2.2 Review of Influential Work in Scene Reconstruction

3D scene reconstruction depends a great deal on the type of imaging system employed. The use of single images, stereo pairs, triplets or image streams from video, all require a different approach. These, in turn, depend on the amount, or lack, of camera calibration information available. It has been shown [Fau92, HGC92] that,

> “in the absence of any constraints, structure can only be recovered up to a projective ambiguity (which differs from true, *Euclidean*, structure by some unknown 3D projective transformation) from a pair of uncalibrated views”.

The “type” of reconstruction can, therefore, be classified according to the ambiguity of the resulting model from the true, Euclidean structure. Mundy and Zisserman [MZ93] showed that the overall accuracy of reconstruction is related to the groups of projective, *planar* transformations [SK52] and developed a framework for specifying the ambiguity resulting from various types of imaging constraints and projective invariants. Table 2.1, has been reproduced from that shown in [HZ00] and lists the four groups of projective transformations:

Group and dof	*Matrix*	*Invariant Properties*
Projective, 15 dof	$\begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$	only the *cross-ratio* of lengths and a few geometrical relationships such as intersection, tangents, inflections and the sign of Gaussian curvature.
Affine, 12 dof	$\begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	parallelism, ratios of areas and lengths on collinear and parallel lines.
Similarity, 7 dof (or metric)	$\begin{bmatrix} s\mathrm{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	all of the above and the ratio of length and angle.
Euclidean, 6 dof	$\begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	inherits all the above plus volume.

Table 2.1: Hierarchy of Projective Transformations.

where A is an invertible 3×3 matrix, R is a 3D rotation matrix, $\mathbf{t} = (\, t_x,\ t_y,\ t_z\,)^T$ is a 3D translation, $\mathbf{v}$ is a general 3-vector, v a scalar and $\mathbf{0} = (\, 0,\ 0,\ 0\,)^T$ is a null vector. It should be noted that, while the subsequent transformations in the table inherit the invariants from those preceding the converse is not true. These four groups of projective transformations define the stratification of 3D geometric structure [Pol00] where Projective is the most general form and Euclidean the most constrained. These terms are also used to describe the type of reconstruction or its ambiguity, except for similarity which equates to *metric*, or scaled-Euclidean, reconstruction. The cross-ratio of four collinear points is defined as:

$$cross(\tilde{\boldsymbol{x}}_1,\ \tilde{\boldsymbol{x}}_2,\ \tilde{\boldsymbol{x}}_3,\ \tilde{\boldsymbol{x}}_4) = \frac{|\tilde{\boldsymbol{x}}_1 - \tilde{\boldsymbol{x}}_2|\,|\tilde{\boldsymbol{x}}_3 - \tilde{\boldsymbol{x}}_4|}{|\tilde{\boldsymbol{x}}_1 - \tilde{\boldsymbol{x}}_3|\,|\tilde{\boldsymbol{x}}_2 - \tilde{\boldsymbol{x}}_4|} \tag{2.1}$$

where $\tilde{\boldsymbol{x}}_i$ are the vertices on a line in homogeneous coordinates for either 1D, 2D or 3D space, thereby making it projectively invariant.

Table 2.1 clearly demonstrates the complexity of the problem. The camera, assuming the pinhole model[2], performs the most general projective transformation, yet the ideal result is the construction of a full, Euclidean, model from which real-world measurements can be extracted. Projects in this area of computer vision research therefore aim to reduce the ambiguity of the reconstructed model as far as possible, given a set of constraints imposed by the particular imaging system employed and the nature of the imaged environment.

Applying constraints to the problem of 3D reconstruction is an inherent part of any solution. Over the last decade, the most obvious constraint, the calibration of the camera, has been studied in great depth. If a camera system is uncalibrated, then the observed structure, from any given image pair, can only be recovered up to a projective ambiguity, as quoted above. However, if additional constraints are available, this projective ambiguity can be reduced so that an affine or a metric reconstruction becomes possible. Luong, Maybank and Faugeras [FLM92] showed that an affine reconstruction can be achieved if three or more views of a static 3D structure are acquired by the same camera in general motion with arbitrary pose and introduced the term *self-calibration*. The concept of camera calibration without capturing an image of a calibration target placed in the world was presented at the same time by Hartley [Har92]. Subsequently this technique has been refined for stereo image pairs with known relative motion, or separation, between the views and is often referred to as *epipolar or auto-calibration*. Examples with pure translation [MvGvDP93, PH95], pure rotation [Har94] and known general transformation [HMDB95, HC98], have been reported and reviewed [But97, Zha98, TM97]

[2]There are a number of possible camera models - the parallel projection, or *affine*, camera model assumes parallel light rays, not necessarily orthogonal to the image plane (as in orthographic), where the projection centre is at infinity. This is of particular use when the objects and/or scene depth are very small

in some depth.

All of the techniques mentioned above approximate the camera projection matrix by first estimating the *Fundamental Matrix*, that defines the epipolar geometry between two views, as described in Appendix A. However, without the addition of other constraints, these techniques can only recover sufficient information for reconstruction *up to scale*. They can not yield the true, Euclidean, structure of the scene. Only with the inclusion of some "known" data, in the *world* coordinate frame can Euclidean reconstruction be achieved. It should be noted, however, that Euclidean reconstruction is not always required and projective reconstructions are often adequate for many applications. The desire for a Euclidean reconstruction requires calibration by the more traditional method of using some *known* calibration pattern, or grid, placed in the imaged scene. The most popular implementation of this was reported by Tsai [Tsa87] but various refinements have since been made [Fau93, HZ00]. Unfortunately, these methods suffer from the obvious limitation of requiring a calibration object and therefore require some "off-line", "prior-to-use" processing. They are often applied to each camera, of the stereo pair, separately and therefore fail to make use of the strict epipolar constraint of two view geometry which is the most fundamental aspect of stereo processing.

Most, if not all, of the reconstruction systems developed to date are intended for a particular application which dictate the choice of imaging system and subsequently the reconstruction techniques employed. Historically, there were only two basic options for the image capture system; calibrated stereo heads, or a monocular camera in motion. The choice of which to use was governed to a large extent by the cost of the hardware or the complexity of calibration. In general, the earlier systems concentrated on the reconstruction of specific objects or scenes, using images captured from positions surrounding the point of interest. These can be referred to as "small-scale" reconstructions, as defined on page 4 of Chapter 1 . The limitation to small-scale reconstruction stemmed partly from the processing capacity of the systems but also, more importantly, from a lack of efficient coverage of the scene, except in cases where large, expensive *scan* imaging equipment is used.

In the last decade, or so, a third type of imaging system has emerged which attempts to address the problem of "where to look next" by scanning the entire scene, effectively to "look every where". Some of these "alternative" stereo imaging systems create *omni-directional* stereo by creating panoramic views, or mosaics, using special mirror systems, or prisms [MYI89, YK90, GG93, PBEP01]. Others form a "virtual" stereo system using a monocular camera rotating on a turn table [IYT92, MB92, KS97]. The specific imaging geometry of these systems will be discussed in Chapter 3. It is the author's opinion that these systems are, in general, either "too complex", "too expensive", or have not completely solved all of the extra processing problems incurred by such systems. This opinion is not unsupported by the referenced author's themselves who claim that the analysis of some methods of omni-directional stereo (fish eye lens, spherical mirror) "are rather local, in the sense that they have concentrated on the problem of acquiring 3D information based on the motion stereo method and much attention has not paid to how we plan the next observation"[IYT92]. This view is supported in [MB92] who insists that "the change in viewpoint must involve a translation of the optical centre." However, [PBEP01] claims that "capturing panoramic omni-stereo images with a rotating cameras makes it impossible to capture dynamic scenes." The author of this dissertation disputes this last claim since, if the angular velocity of the rotating camera is kept constant, then any structure with an appropriate motion vector is static whereas any structure with a greater motion vector is obviously in motion. The capture and reconstruction of dynamic scenes with a camera in motion is therefore not only possible but has already been demonstrated in [TM93].

Unlike most of these previous examples of alternative stereo systems, periscopic stereo does not concentrate on producing a full panoramic view, it simply uses the inherent geometry of a panoramic scan of the surrounding environment, acquired as the natural consequence of rotating a periscope, at a fixed velocity, above a central optical axis. As stated earlier, one of the advantages of periscopic stereo is that it is able to make use of many existing techniques with “relatively” minor modifications.

Now that most of the basic concepts have been identified, the following is a brief review of three major projects, two of which are directly relevant to this area of research. These projects have been chosen because they are, in the opinion of the author of this dissertation, most responsible for extending the boundaries in this area of computer vision research and will continue to be the primary source of reference in the future.

2.2.1 Review of Major Research Projects

There have been a number of recent international workshops devoted specifically to the techniques associated with scene reconstruction and 3D computer vision. The most notable were;

- 2nd Joint European–US Workshop on “Applications of Invariance in Computer Vision”, Ponta Delgada, Azores, October 1993. [MZF93]
- Int. NSF–ARPA Workshop on “Object Representation in Computer Vision”, New York City, NY, USA, December 1994. [HPBG94]
- Int. Workshop on “Object Representation in Computer Vision II”, Cambridge, U.K., April 1996. [PZH96] (in conjunction with ECCV96).
- European Workshop on “3D Structure from Multiple Image of Large–Scale Environments (SMILE’98)”, Freiburg, Germany, June 1998. [KvG98] (in conjunction with ECCV98).
- IEEE Workshop on “Multi-View Modeling and Analysis of Visual Scenes”, Colorado State University, Fort Collins, USA., June 1999. [KS99] (in conjunction with CVPR99).

Of particular importance is the 1998 European ‘SMILE98’ workshop [KvG98] which brought together researchers from the most prominent projects in this area of research. The following is a brief description of the three main projects represented at SMILE'98 workshop. A complete description of these projects is given in the introduction chapter of the workshop's proceedings [KvG98].

VANGUARD - (Visualization Across Networks using Graphics and Uncalibrated Acquisition of Read Data) aimed to automatically create realistic 3D models for use in AR and VR applications from a single "uncalibrated" video camera moving in unknown and unconstrained motion. The applications selected for the project focused on four specific areas; '3D Surface Modeling' to improve geometric modelling techniques and produce more realistic models, 'Collaborative Scene Visualization' where CAD modelled objects are combined with 3D models extracted from imagery to create dynamic virtual environments, 'Tele-exhibitions for museums' where all the exhibits and museum interior are reconstructed in such a way to allow fully interactive "walk throughs" over the Internet and finally 'Stereo Visualization of objects and scenes' from monocular image sequences to create 'pseudo-holographic' displays. These applications required drawing together expertise from both Computer Graphics and Computer Vision and were only made possible by classifying objects within the scene and extracting both geometry and surface descriptions specific to these classifications. Three types of objects were extracted and modelled; 'Individual objects' for which a full model was required, 'room interiors' for which only a part models were required and 'natural outdoor scenes' which are largely planar textures. The project has now concluded but some related work still continues at Oxford, where the project's website is located (at `http://www.robots.ox.ac.uk/~vanguard/`).

CUMULI - (Computational Understanding of Multiple Images) is a long term ESPRIT project focusing on multi-image geometry and its application to 3D industrial metrology. It is effectively a follow up project to the ESPRIT-BRA project VIVA. The objectives, building on the insights into the geometry of 3D perception provided by VIVA, are three-fold. Firstly the aim is to recover 3D structure under three different situations; ‘Unknown camera parameters and scene’ which has been well studied but lacks unified theory, ‘Partial camera or scene knowledge’ where *a-priori* image cues are used to extend the range or quality of reconstruction from minimal image data and the correspondence of ‘non point-like image features’ such as lines, curves and planes where the stronger geometric constraints allow relaxed conditions for reconstruction and improve accuracy. Recent advances in multi-camera geometry [HÅ97, Tri97b], auto-calibration [HÅ96, Tri97a] and efficient reconstruction [Spa96] have all been reported. Secondly is to study the concepts of 3D perception in terms of the image sequence and non-rigid motion estimation. Although the same underlying theory applies, the fact that under certain circumstances the geometry inherited from the discrete case becomes degenerate leads to a requirement for more appropriate incremental geometry to be developed. This also includes the study of multi-image matching constraints, the development of better tracking tools and more efficient reconstruction methods for the continuous recovery of large-scale environments. Lastly the generation of automated algebraic and geometric reasoning tools which are required for the construction of complex large-scale models for AR and VR applications. This is a new area of research which suggests that geometric models should be more than just textured, rigid wire frames and should be more dynamic and incorporate real-world constraints. This project is ongoing and more information can be found at: (`http://www-sop.inria.fr/robotvis/projects/`)

PANORAMA - ACTS PANORAMA Project consists of a consortium of 14 European partners from various universities, research institutes and industry and concentrates on the enhancement of visual information exchange in telecommunications with 3D telepresence. The system under development makes use of a calibrated trinocular camera system, an autostereoscopic display, real-time image processing using special purpose hardware which enables the creation of dynamic and photo-realistic models for video conferencing. The system incorporates techniques for fusing of 3D model data with image textures and computer generated graphics to create the photo-realistic models. This image synthesis approach also generates interpolated, intermediate views using vector coding techniques which smooths out motion effects of the deformable 3D models. Details of the project can be found at: `http://www.tnt.uni-hannover.de/project/eu/panorama/`

'VANGUARD' and 'CUMULI' are regarded as being particularly relevant to the research presented in this dissertation. The 'VANGUARD' project provides an insight to the level of accuracy acceptable for 3D reconstructions for VR "walk-through" applications and has successfully demonstrated the use of single camera systems. However, Euclidean reconstruction is not possible with this system, as described in Section 2.2. The 'CUMULI' project provides advances in the description of multi-camera geometry and calibration which are more directly relevant to periscopic stereo and mobile robot applications. 'PANORAMA' is not particularly relevant to the research presented in this dissertation, but has been mentioned to complete a picture of the "state of the art" of associated technology.

Although both the 'VANGUARD' and 'CUMULI' claim to address (directly or indirectly) large-scale 3D scene reconstruction neither have been "fully" demonstrated. The examples given by both projects, available via their respective web sites, are impressive but are only sections of large-scale reconstructions. The author of this dissertation has not, as yet, found examples of navigable (in both a VR and a robotic sense), large-scale environments. This may be due, in part, to commercial considerations. However, the fact that neither of these projects addresses the fundamental problem of "where to look next", suggests that these and many other similar projects have a limited capability of producing large-scale, VR style, environments.

As suggested at the beginning of this chapter, there is far more literature on the subject of 3D reconstruction than can be presented here. The above is therefore only a cursory review of increasingly expanding area of computer vision research. A list of people, and/or collaborating institutions, on the projects given above, together with a short list of some other notable reconstruction projects, is given in Appendix B.

2.3 Choice of Image Processing and Computer Vision Software Framework

Before any useful results can be obtained from experiments into computer vision techniques a software environment, or framework, is required to house the various types of image processing and display tools. Even if this framework amounts to little more than an *ad-hoc* collection of programs, the interaction between them is extremely important and will have a considerable effect on the productivity. There are two obvious choices, "off-the-shelf" or "writing your own". The author of this dissertation expended considerable time on the assessment of such frameworks and Appendix C provides an insight to the considerations required when choosing or writing your own framework.

The software framework ultimately chosen to support the research presented in this dissertation was `Tina`[3]. The reasons for this choice are as follows:

Firstly, `Tina` is written in 'C' which provides, in the opinion of the author, the best compromise between the application of object oriented design (which is possible in 'C') and the flexibility to define software models and programming structures required by most, "state of the art", computer

[3]The source code, complete with documentation and examples, is available for research use on public license and can be downloaded from: `http://www.niac.man.ac.uk/Tina/index.html`

vision techniques. It has a simple graphical user interface which offers the possibility of constructing scripted algorithms, as well as the easy construction of button activated processing. There is a considerable amount of functional decomposition designed into the source code at all levels allowing for the maximum amount of code reuse and the possibility to developing new algorithms quickly and efficiently. The central core of the system is designed around a stack memory model which allows simple transfer of image data between various processing tools. This allows for the comparison of algorithms with efficient 'reload' data and 'undo' functionality. Each tool is however free to create new image data, or make copies, as required. Access to the image data is extremely flexible, allowing complete random access down to pixel level, but via consistent *get* and *put*, 'data-type' invariant methods. Many of these are implemented by efficient, in-line, macro's. The data structures are well conceived and offer good flexibility. In particular most of the image data objects contain a *properties* list which allow for the dynamic extensions of, and associations between, data objects. A full set of tools for the usual forms of dynamic data structures, such as, single-ended, double-ended and recursive linked lists, trees and graphs, are also included. The framework contains a sufficient set of mathematical, image processing and computer vision libraries, as well as useful graphics and display tools, which are appropriate for 3D reconstruction tasks. Finally this framework, unlike many of its contemporaries, is a small, lightweight package, consisting only of the more essential components. This greatly simplifies its use. `Tina` is not, however, perfect and does have a few short comings. Although the documentation is quite good, the code itself is poorly commented in places and contains the occasional bug. In the opinion of the author of this dissertation, `Tina` is probably the most flexible image processing framework freely available and does not incur excessive effort before productive research can be realized.

2.4 Summary

This chapter provides background information for the concepts and terminology used in this dissertation. Using Marr's bottom-up approach [Mar82] four processes are identified which constitute 3D reconstruction. A pre-process, feature detection stage provides the basic information for three mutually dependent processes. These are; stereo or correspondence matching, stereo camera calibration and the projection of disparity images or reconstruction. Aspects of all of these are covered in this dissertation. A final post-processing stage, for the production of visually realistic models, is not covered in the research presented. Technical reviews of these subjects are deferred to the specific chapters.

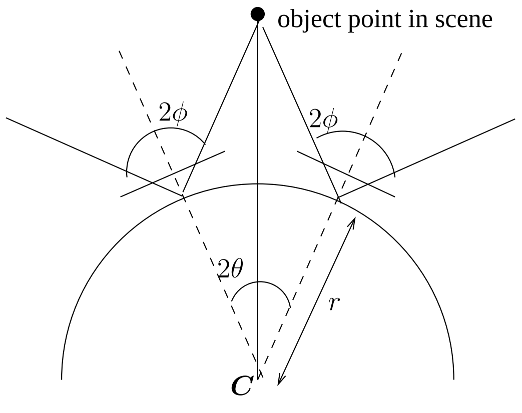
Two research projects have been identified which have provided both impetus for the research presented here and a source of reference for the latest techniques in 3D reconstruction. An image processing and computer vision framework has been reviewed since this also provides a source of proven techniques.

Chapter 3

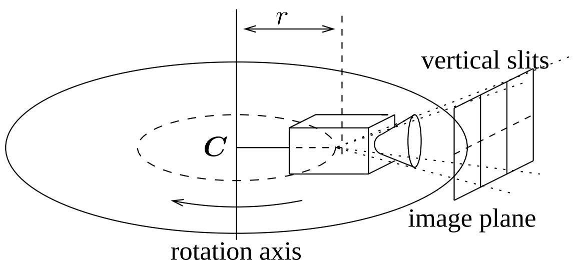
Periscopic Stereo Vision

In Chapter 2, the concept of implementing a stereo imaging system with a single camera, was identified as a possible solution to the problem of efficient capture of the surrounding scene. A number of research projects have supported the concept of "alternative" stereo imaging systems in recent years but, in general, most of the systems proposed to date have introduced as many problems as they claim to have solved [MYI89, YK90, IYT92, MB92, GG93, Sze96, PH97, PBEP01]. Not all of these have been directly concerned with 3D reconstruction. However, each one has attempted to capture sufficient information from the image data to implement various forms of robot navigation. An early example presented in [MYI89] used a camera with a fish-eye lens mounted on a mobile robot and recovered depth of vertical structure in the scene from a sequence of images, captured while in motion. Apart from improving the field of view the system provided no performance advantage over a standard single camera in motion to outweigh the complexity of image processing which required a spherical mapping to convert the distorted image created by the fish-eye lens. Other systems have used the reflections from a conic mirror [YK90], a split mirror [GG93] and only recently spiral (more actually described as a dome) mirrors and spiral lens [PBEP01]. With the exception of [GG93], all these systems required specialized feature detection and correspondence algorithms which add considerable complexity for advantage of a panoramic view which ultimately yields low accuracy depth information due to the lack of either any translation of the camera's optical center or small baseline. However, one major advantage of some of these systems is that they are entirely self-contained with no external mechanics. While these imaging systems have been applied to robotic navigation with varying degrees of success, as yet, there are no successful application to 3D reconstruction.

The use of rotation to achieve *omni-directional* stereo was first, successfully, demonstrated in [IYT92]. This work and later examples [Sze96, PH97] rely on rotating the camera on a turn table. In [IYT92] the panoramic view is created by capturing the scene through two vertical slits mounted on the turn table in front the of the camera, as shown in Figure 3.1. This has the obvious disadvantage of requiring



(a) imaging geometry

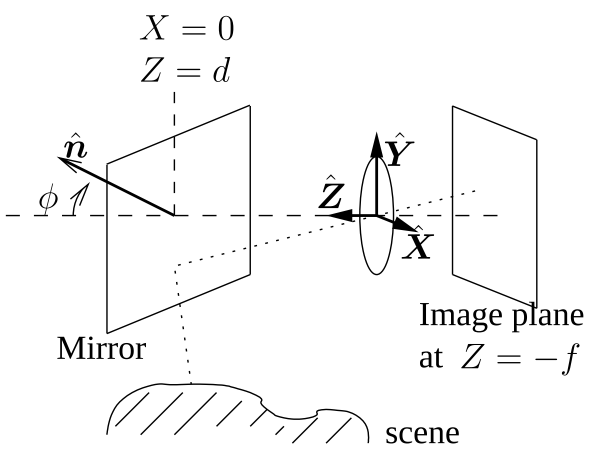


(b) rotating camera

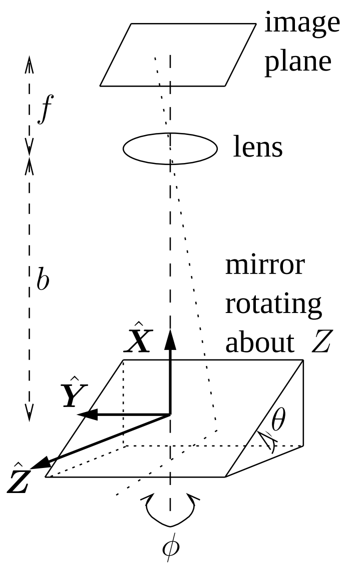
Figure 3.1: Sketch of rotating turn table camera with imaging geometry.

large, cumbersome, equipment. It also introduces a complication to camera, or system, calibration due to a requirement for accurate positional feedback for the relative displacement of the camera. However, the recovery of depth information from these rotating camera systems is much better than the static omni-directional systems since the baseline distance is greater and can often be varied to provide improved performance for near or distant scenes. Apart form reference to [IYT92] and [Sze96, PH97], which have a similar imaging geometry, Figure 3.1 has been reproduced here for visual comparison of these systems and the genealogy of periscopic stereo. Note in Figure 3.1 that these systems create a *divergent* stereo view.

The merits and demerits of the omni-directional and rotating camera systems are in direct opposition. However, a system that combined the merits of both would be an attractive alternative, especially for large-scale scene reconstruction. The idea of implementing a stereo vision system using a single, flat, rotating mirror was first reported by Murray and Beardsley [MB92]. In their system the axis of rotation of the mirror was perpendicular to the optical axis of the camera, as shown in Figure 3.2(a). With this configuration imaged objects would appear to move horizontally across the image plane for a given sequence. This arrangement yields a visually (human) sensible data set but it relies on a somewhat complicated mechanical arrangement and an accurate measurement of the angle of the actuated mirror plane for the camera geometry. Murray and Beardsley [MB92] suggested a second,



(a) horizontal scanning mirror from [MB92]



(b) rotating mirror from [CC94b]

Figure 3.2: Sketches of horizontal scanning and rotating cameras.

simplified, mirror arrangement where the axis of rotation lies along the optical axis of the camera, as shown in Figure 3.2(b). However, they declared its use to be impractical due to the “tumbling” effect induced on the image data by the trajectory of the *virtual* camera which rotates in sympathy with the mirror. Clark and Chan [CC94b] adopted this second mirror arrangement and applied simple geometric transformations on the image data to return, or “untumble”, it to a visually sensible, horizontally scanned sequence. They concluded that this arrangement would in fact offer many advantages over existing systems and introduced the term *periscopic stereo*. Both of these configurations implement a *convergent* stereo view from successive frames of the image sequence captured as the virtual camera rotates in sympathy with the mirror.

While the difference in the imaging geometry of these systems is subtle, the implementation of the image processing for the respective configurations is significant. In the latter case [CC94b], the configuration proposed retains all the merits of its predecessors without, reportedly, incurring excessive complexity. However, the concept of “tumbled” and “untumbled” image data is a continuous thread throughout this dissertation and is the fundamental reason for the system and image processing analysis contained herein. This chapter reviews the concept periscopic stereo and introduces some practical points on the construction and operation of a realizable system. The analysis of the virtual, relative, camera geometry has been reworked here, since the version offered in [CC94b] was for a system which was mounted upside down and assumed a left handed coordinate system, in order to allow for suitable alignment of the camera, mirror and world coordinate systems that would be applied in the real system with the camera pointing vertically up.

3.1 Virtual Camera Geometry

Periscopic stereo is essentially a system which implements a stereo vision system by using a mirror that rotates about the camera's optical axis, as shown in Figure 3.3. This system creates a series of virtual camera positions where any two frames from a sequence form a converging stereo view of the scene. The origin of the system is defined as the centre of the mirror plane and a *right-hand* coordinate system has been chosen in order to simplify the use of existing software within the `Tina` framework. The analysis for a left-hand coordinate system would be the same except for the interchange of some elements in the vector and matrix representations.

The real camera $\boldsymbol{C}_r$ is situated at some distance b along the axis labelled Y_m from the centre of the mirror plane $\boldsymbol{C}_m$. The centre of the mirror plane is assumed to be the system origin and is labelled $\boldsymbol{O}_m$. This choice of system origin is fundamental and its importance is explained later in the analysis of depth estimation from periscopic stereo in Section 3.3. The choice of X_m, Y_m and Z_m axis are consistent with a right-handed coordinate system with Z_m axis pointing out into the scene. This is in keeping with with standard theory [SHB99].

With reference to Figure 3.3, the basic geometry of the system can be derived by inspection. The surface normal $\hat{\boldsymbol{n}}$ to the mirror plane $\boldsymbol{C}_m$ is given by:

$$[\,\sin\theta\ \sin\phi\,, \cos\theta\,, \sin\theta\ \cos\phi\,]^T \tag{3.1}$$

and position of the mirror plane, with respect to the camera, can be derived from, $\alpha = b\ \cos\theta$ (the translation from the real camera position to the mirror plane) and its normal such that:

$$\boldsymbol{C}_m = \boldsymbol{C}_r - \alpha\,\hat{\boldsymbol{n}} \tag{3.2}$$

This can be thought of as a simple perpendicular projection of the real camera onto the mirror plane.

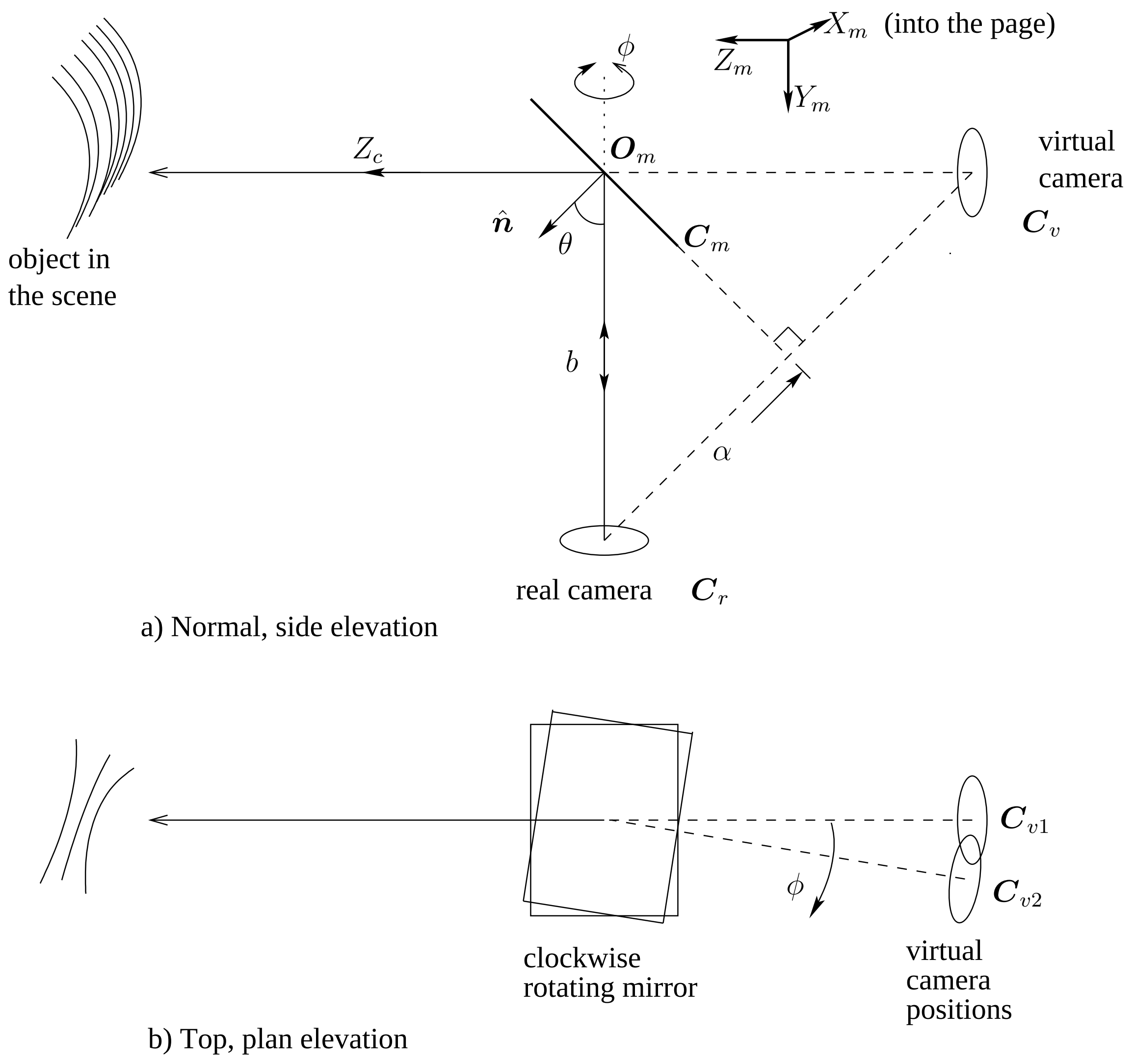


Figure 3.3: Basic geometry for the periscopic stereo head.

The position of the virtual camera is therefore given by:

$$\boldsymbol{C}_v = \boldsymbol{C}_r - 2\alpha\,\hat{\boldsymbol{n}} \tag{3.3}$$

Using standard trigonometric relationships Equation 3.3 simplifies to:

$$\boldsymbol{C}_v = \begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix} - 2b\cos\theta \begin{bmatrix} \sin\theta\,\sin\phi \\ \cos\theta \\ \sin\theta\,\cos\phi \end{bmatrix} = -b \begin{bmatrix} \sin 2\theta\,\sin\phi \\ \cos 2\theta \\ \sin 2\theta\,\cos\phi \end{bmatrix} \tag{3.4}$$

If we assume that $\theta = 45^\circ$ the respective terms in Equation 3.4 disappear, demonstrating that as the mirror rotates the position of the virtual camera tracks around behind it in the plane $Y_m = 0$. Other reasons for eliminating θ will be explained later in Section 3.3.

Only the position of the centre of the virtual camera has thus far been determined. The analysis must be extended to determine the behaviour of all points on the image plane. Considering some arbitrary point $\boldsymbol{p}_r$ "near"[1] which is offset from the optical centre along the $x-$, $y-$ and $z-$axes such that, in general:

$$\boldsymbol{p}_r = [\,\delta x\,, b + \delta y\,, \delta z\,]^T \tag{3.5}$$

The virtual position of this point can be determined from Equation 3.3 with a new translation scalar, α_n, for the projection across the mirror plane, as shown in Figure 3.4, such that:

$$\boldsymbol{p}_v = \boldsymbol{p}_r - 2\alpha_n\,\hat{\boldsymbol{n}} \tag{3.6}$$

As before, α_n is determined from the origin of the point in question and the surface normal to the mirror plane by considering that $\boldsymbol{C}_m \cdot \hat{\boldsymbol{n}} = 0$ and substituting for $\boldsymbol{C}_m$ in Equation 3.2 such that:

$$(\boldsymbol{C}_r - \alpha\hat{\boldsymbol{n}}) \cdot \hat{\boldsymbol{n}} = 0 \tag{3.7}$$

Given $\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{n}} = 1$, yields $\alpha = \boldsymbol{C}_r \cdot \hat{\boldsymbol{n}}$, or, by the same argument, for the new point:

$$\alpha_n = \boldsymbol{p}_r \cdot \hat{\boldsymbol{n}} = \sin\theta\,\sin\phi\,\delta x + \cos\theta\,(b + \delta y) + \sin\theta\,\cos\phi\,\delta z \tag{3.8}$$

[1]The use of *near* is to allow for a non-zero δy which is required for the transposition of the axis due to the reflection component as shown in Figure 3.4

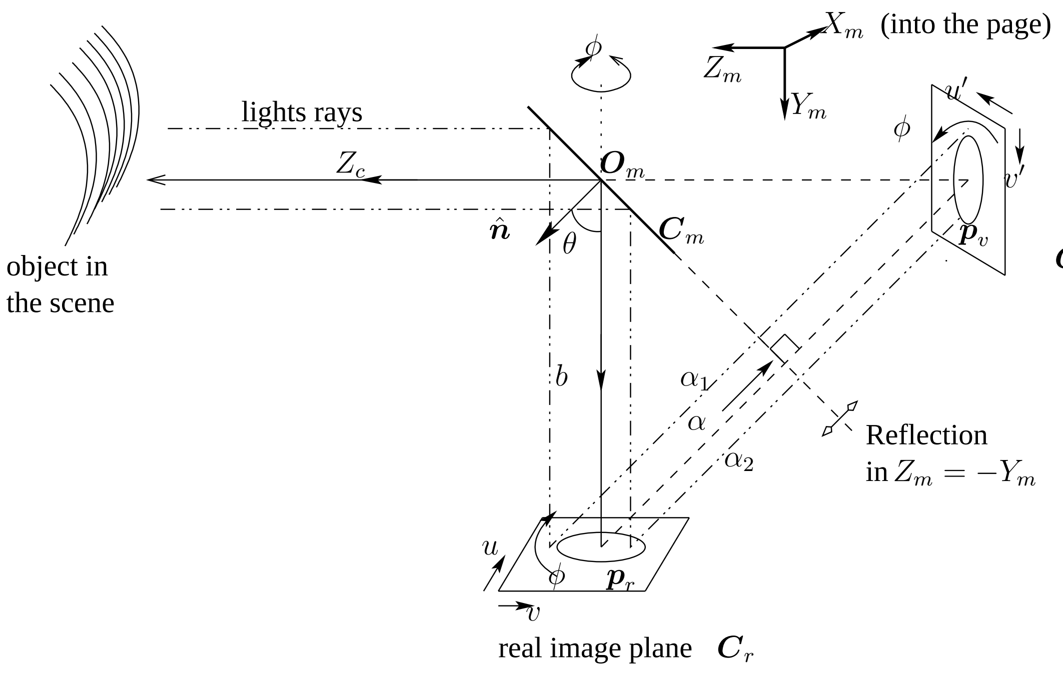


Figure 3.4: Reflection and rotations in periscopic stereo.

Substituting for Equation 3.8 in Equation 3.6 yields,

$$
\begin{aligned}
\boldsymbol{p}_v &= \boldsymbol{p}_r - 2\,(\boldsymbol{p}_r \cdot \hat{\boldsymbol{n}})\,\hat{\boldsymbol{n}} \\
&= \begin{bmatrix} \delta x \\ b+\delta y \\ \delta z \end{bmatrix} - 2\,(\sin\theta\,\sin\phi\,\delta x + \cos\theta\,(b+\delta y) + \sin\theta\,\cos\phi\,\delta z) \begin{bmatrix} \sin\theta\,\sin\phi \\ \cos\theta \\ \sin\theta\,\cos\phi \end{bmatrix} \\
&= \begin{bmatrix} (\,1 - 2\sin^2\,\theta\,\sin^2\,\phi\,)\,\delta x - \sin 2\theta\,\sin\phi\,(b+\delta y) - 2\sin^2\,\theta\,\sin\phi\,\cos\phi\,\delta z \\ -\sin 2\theta\,\sin\phi\,\delta x + (\,1 - 2\cos^2\theta\,)(b+\delta y) - \sin 2\theta\,\cos\phi\,\delta z \\ -2\sin^2\,\theta\,\sin\phi\,\cos\phi\,\delta x - \sin 2\theta\,\cos\phi\,(b+\delta y) + (\,1 - 2\sin^2\,\theta\,\cos^2\,\phi\,)\,\delta z \end{bmatrix} \quad (3.9)
\end{aligned}
$$

using the double angle, trigonometric relationships for $\sin 2\theta$ and $\cos 2\theta$ where necessary. Equation 3.9 simplifies to Equation 3.4 when $\delta x = \delta y = \delta z = 0$.

A comparison of the terms in the positional vectors derived in Equations 3.4 and 3.9 reveals that, even ignoring the effect of θ, extra ϕ and ϕ^2 terms have appeared in the latter. This demonstrates that any point on the image plane, except the point coincident with the optical axis, will be subject to a tumbling motion consisting of two rotational components. The first rotational component is the circular track of the virtual camera about the mirror centre, or system origin, and the second is a rotation of the virtual image plane about its optical axis. Both due to the effect of ϕ.

Although the system is centered about the mirror plane, none of the above analysis has been concerned with identifying a specific reflection component. While this reflection component is present in Equations 3.4 and 3.9 it has been deliberately ignored, until now, in order to simplify and aid description of the system geometry.

In order to isolate and remove the components which induce the tumbling motion of the image data it is necessary to analyze the 3D spatial transformation which maps any point on the real image plane onto its corresponding point on the virtual image plane. Assigning the spatial transformation matrix, T, and rewriting Equations 3.4 and 3.9 as a pair of simultaneous equations yields:

$$\begin{aligned} \boldsymbol{C}_v &= \mathrm{T}.\,\boldsymbol{C}_r \\ \boldsymbol{p}_v &= \mathrm{T}.\,\boldsymbol{p}_r \end{aligned} \tag{3.10}$$

Subtracting to give $\boldsymbol{C}_v - \boldsymbol{p}_v = \mathrm{T}.\,(\boldsymbol{C}_r - \boldsymbol{p}_r)$ and substituting for $\boldsymbol{C}_v$, $\boldsymbol{C}_r$, $\boldsymbol{p}_v$ and $\boldsymbol{p}_r$ yields,

$$\mathrm{T}\cdot\left\{\begin{bmatrix}0\\ b\\ 0\end{bmatrix} - \begin{bmatrix}\delta x\\ (b+\delta y)\\ \delta z\end{bmatrix}\right\} = \left\{-b\begin{bmatrix}\sin 2\theta\,\sin\phi\\ \cos 2\theta\\ \sin 2\theta\,\cos\phi\end{bmatrix} - \begin{bmatrix}\\ \boldsymbol{p}_v\\ \\ \end{bmatrix}\right\} \tag{3.11}$$

which simplifies to:

$$
\mathrm{T}\cdot\begin{bmatrix} -\delta x \\ -\delta y \\ -\delta z \end{bmatrix} = \begin{bmatrix} -(\,1-2s^2\theta s^2\phi\,)\delta x - bs2\theta s\phi + (\,s2\theta s\phi\,)(b+\delta y) + 2s^2\theta s\phi c\phi\delta z \\ +s2\theta s\phi\delta x - bc2\theta - (1-2c^2\theta)(b+\delta y) + s2\theta c\phi\delta z \\ +2s^2\theta s\phi c\phi\delta x - bs2\theta c\phi + (\,s2\theta c\phi\,)(b+\delta y) + (\,1-2s^2\theta c^2\phi\,)\delta z \end{bmatrix}
$$
$$
= \begin{bmatrix} (\,1-2\sin^2\theta\,\sin^2\,\phi\,)\delta x - (\,\sin 2\theta\,\sin\phi\,)\delta y - 2\sin^2\theta\,\sin\phi\,\cos\phi\,\delta z \\ -\sin 2\theta\,\sin\phi\,\delta x - (\cos 2\theta)\delta y - \sin 2\theta\,\cos\phi\,\delta z \\ -2\sin^2\theta\,\sin\phi\,\cos\phi\,\delta x - (\,\sin 2\theta\,\cos\phi\,)\delta y + (\,1-2\sin^2\theta\,\cos^2\phi\,)\delta z \end{bmatrix} \tag{3.12}
$$

where $-bc2\theta - (\,1-2c^2\theta\,)(b+\delta y) \equiv -b\cos 2\theta - (-\cos 2\theta)(b+\delta y)$, switching from shorthand notation and applying alternative double angle trigonometric relationships.

Factorizing to form the transformation matrix yields:

$$
\mathrm{T} = \begin{bmatrix} 2\sin^2\,\theta\,\sin^2\,\phi\,-1 & \sin 2\theta\,\sin\phi & 2\sin^2\,\theta\,\sin\phi\,\cos\phi \\ \sin 2\theta\,\sin\phi & \cos 2\theta & \sin 2\theta\,\cos\phi \\ 2\sin^2\,\theta\,\sin\phi\,\cos\phi & \sin 2\theta\,\cos\phi & 2\sin^2\,\theta\,\cos^2\,\phi\,-1 \end{bmatrix} \tag{3.13}
$$

The factorization in Equation 3.13 is derived by simply isolating the vectors components and rewriting in matrix form. It should be noted that there was a typographical error in the first column, third row of equation (14) in [CC94b], which should read $\sin 2\theta\,\cos\phi$ and not $\sin 2\theta\,\sin\phi$.

The transformation matrix given in Equation 3.13 contains a reflection and the two rotational components identified above. It is apparent from Equation 3.13 that the virtual camera rotates about the optical axis, Z_m, and also about the system's vertical axis, Y_m. Removing the rotation about the optical axis and the reflection of X_m, due to the mirror, would return a sequence of images to a normal, *fronto-parallel*, scan of the horizon. Removing these components effectively "untumbles" the image sequence.

Combining the standard matrix for a clockwise 3D rotation about Y_m and a reflection about the $X_m = 0$ plane yields:

$$\mathrm{CW\,Rot_y}\phi.\mathrm{Rflt_{x=0}} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} = \mathrm{T}_m \tag{3.14}$$

where T_m the matrix transformation due to both components.

The matrix transformation without these components T' can be determined from $\mathrm{T}' = \mathrm{T}_m^{-1}\,\mathrm{T}$ which yields:

$$\mathrm{T}' = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ \sin 2\theta\,\sin\phi & \cos 2\theta & \sin 2\theta\,\cos\phi \\ -\cos 2\theta\,\sin\phi & \sin 2\theta & -\cos 2\theta\,\cos\phi \end{bmatrix} \tag{3.15}$$

Comparison of Equation 3.15 with equation (15) in [CC94b] shows that they have identical components, albeit for an interchange of the first and seconds rows and columns due to the redefined coordinate system. Again, there is a typographical error in equation (15) of [CC94b] where the sign of $\sin 2\theta$ in the first column of the third row is incorrect. Confirmation of Equation 3.15 can be determined by substituting T' back in $\mathrm{T} = \mathrm{T}_m\,\mathrm{T}'$ and using the double angle, trigonometric relationships, as in Equation 3.9 above.

With the squared components of the first and third rows of Equation 3.13 now removed and assuming $\theta = 45°$, the relative camera geometry of the imaging system, described by Equation 3.15, has effectively been reduced to a scan of the horizontal plane about the system axis Y_m. In this configuration any subsequent stereo processing can be conducted along single raster lines in the images, instead of across the whole image plane.

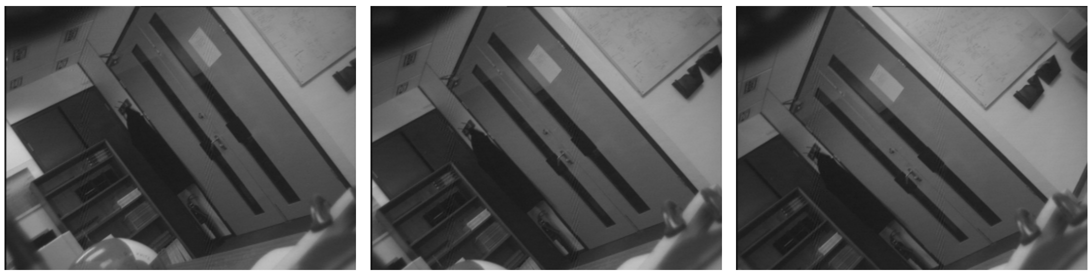
3.2 Removing the Tumbling Motion

In practice removing the tumbling motion from a sequence of image data is relatively straight forward. As each image is recovered from the camera's frame buffer, it can be stored in the images' data array by reading backwards from right to left, thereby implementing a reflection of the x image coordinate. This equates to a reflection in the plane $X_m = 0$, in Figure 3.4, or alternatively a reflection in the plane, $X_c = 0$, in the camera coordinate frame in Figure A.1 of Appendix A.

A rotation matrix, centered about the optical axis, can then be applied with an operator equivalent to the angular displacement between frames. This is achieved using the computer graphics technique of applying a translation to the rotation origin, a 2D rotation about that origin and finally applying an inverse translation to the compensate for the first. This geometric transformation of the image data often yields new pixel coordinates which lie between the original pixel grid. A close approximation of the new intensity value for the pixel is therefore derived by interpolation. However, interpolation effectively adds a re-sampling, or filter, component to this image "pre-processing" stage which is not, in general, desirable [Moh93]. This is discussed further in Chapter 4.

Figure 3.5 shows three frames from the original image sequence and the same frames with the rotation about the optical axis corrected. Correcting for the rotation about the optical axis introduces a rotating frame into the image data, as shown in Figure 3.5(b).This frame, referred to in this dissertation as the *silhouette* frame, complicates subsequent processing. This is discussed in Chapter 5. However, it should be noted that it is not necessary to correct the virtual image plane prior to all subsequent processing. The process of removing the tumbling motion from the image data is, in practice, similar to image plane *rectification*, except that the true *canonical configuration*[2], as shown in Figure 3.6(a), is not completely achieved. This is discussed further in Chapters 5 and 6 later.

[2]The canonical configuration for a stereo head is with parallel optical axes and coplanar image planes


(a) original images

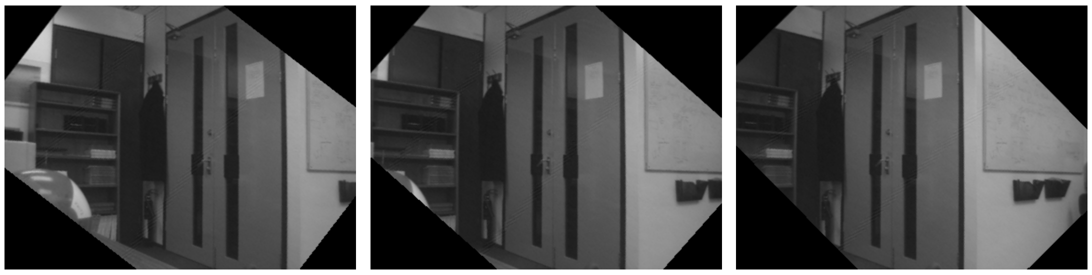
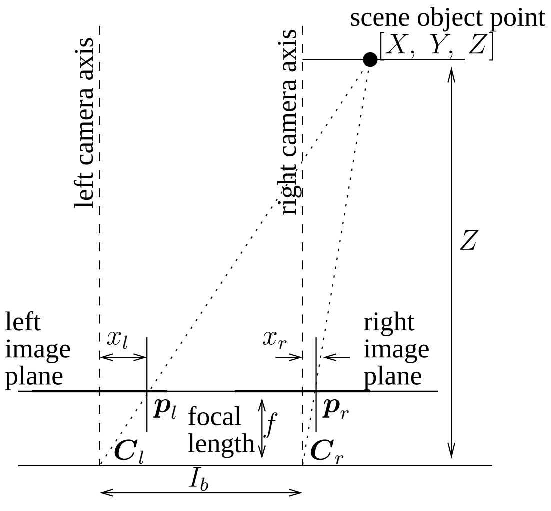
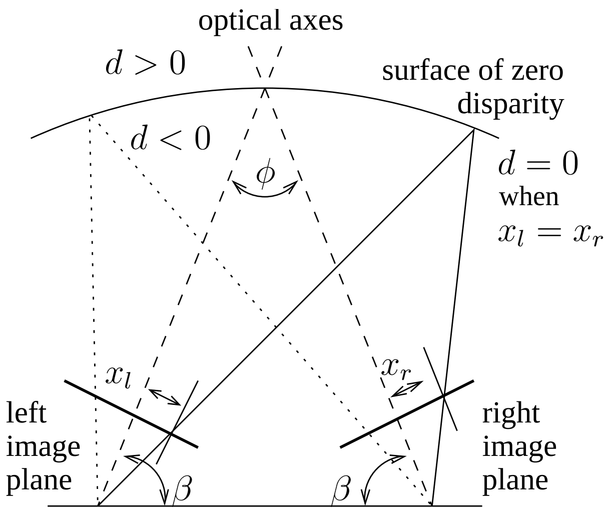

(b) corrected images

Figure 3.5: Sequence of original and rotationally corrected images.



(a) canonical stereo configuration



(b) disparity in convergent cameras

Figure 3.6: Sketches of (a) canonical for stereo and (b) disparity in convergent cameras.

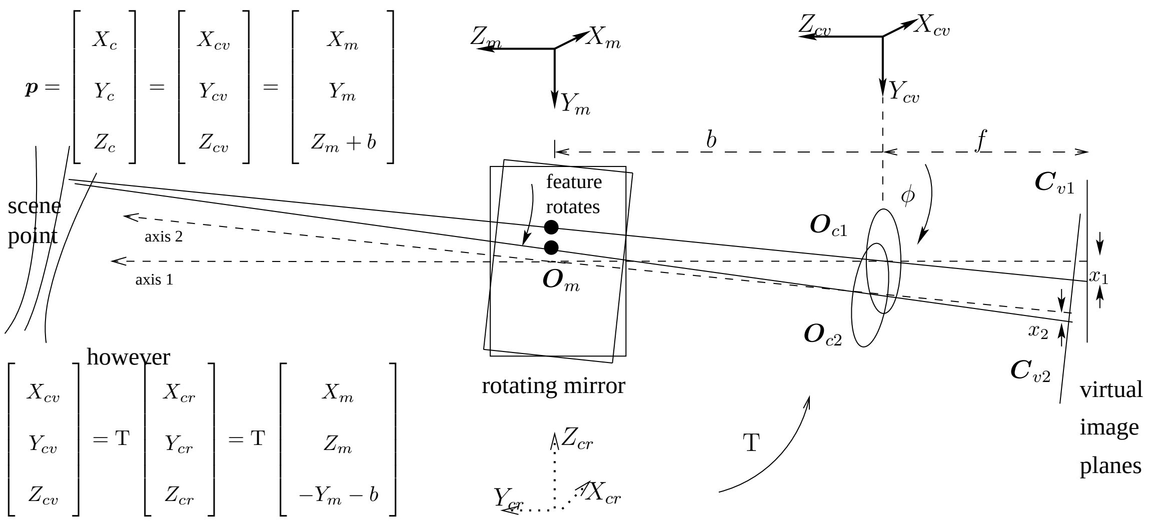
3.3 Depth from Periscopic Stereo

Recovering depth from stereo images requires the estimation of the disparity between corresponding image points. In a canonical stereo head the distance from some 3D point in the scene is determined by:

$$Z = \frac{I_b f}{(x_l - x_r)} \tag{3.16}$$

where I_b is the interocular separation between the cameras, or *baseline*, f is focal length of the camera/s and $x_l - x_r$ is the horizontal disparity between the corresponding images points, as shown in Figure 3.6(a) (reproduced from [JKS95, chap.11] for ease of reference). The geometry of the periscopic stereo head is not as straight forward. However a measure of disparity and hence an estimation of depth are still possible.

Figure 3.7 demonstrates the situation where a scene point is imaged in two, sequential, virtual image planes. First consider the situation where the system origin $\boldsymbol{O}_m$ coincides with a camera rotating, horizontally, about its own optical centre (or the Y_m axis, "into the page"). For a single image mea-



$\boldsymbol{p} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} X_{cv} \\ Y_{cv} \\ Z_{cv} \end{bmatrix} = \begin{bmatrix} X_m \\ Y_m \\ Z_m + b \end{bmatrix}$
however
$\begin{bmatrix} X_{cv} \\ Y_{cv} \\ Z_{cv} \end{bmatrix} = \mathrm{T} \begin{bmatrix} X_{cr} \\ Y_{cr} \\ Z_{cr} \end{bmatrix} = \mathrm{T} \begin{bmatrix} X_m \\ Z_m \\ -Y_m - b \end{bmatrix}$
Z_m
X_m
Y_m
Z_{cv}
X_{cv}
Y_{cv}
b
f
scene point
axis 2
axis 1
feature rotates
$\boldsymbol{O}_m$
ϕ
$\boldsymbol{O}_{c1}$
$\boldsymbol{O}_{c2}$
$\boldsymbol{C}_{v1}$
$\boldsymbol{C}_{v2}$
x_1
x_2
rotating mirror
Z_{cr}
Y_{cr}
X_{cr}
T
virtual image planes

Figure 3.7: Depth from periscopic stereo.

surement, the scene point is related, via similar triangles from the perspective, or central, projection model of a camera system (see Appendix A, Equation A.3) to the world point by,

$$m_x = \frac{x}{f} = \frac{X_c}{Z_c} \qquad \text{or} \qquad m_y = \frac{y}{f} = \frac{Y_c}{Z_c} \tag{3.17}$$

By rotating the camera by fixed, angular displacements, ϕ, multiple instances of this are generated and depth is derived from the disparity of the image points and the focal length of the camera. Now consider the virtual camera position displaced along the Z_m axis by I_b. Rotating the mirror now generates a series of virtual cameras each displaced by an interocular separation determined by the angle of rotation and the distance I_b. These parameters effectively determine the stereo baseline. Subsequent camera positions now converge onto the system origin introducing a limitation to the sign of the disparity. It is known that converging cameras yield an arc of zero disparity with a radius determined by the product of the convergent optical axes [JKS95], as shown in Figure 3.6(b). Only 3D points beyond the arc are valid and therefore will always generate a positive measure of disparity, assuming a calculation using left camera to right camera's imaged points.

Using the disparity from corresponding imaged points together with the relative orientation between the camera positions, an estimate of the 3D position of a point in the scene can be derived. However, the position of the virtual camera is determined by the spatial transformation T' (ignoring the effects of T_m) and specified in terms of the system's coordinate frame about $\boldsymbol{O}_m$. Applying the change in coordinate system and the transformation, as shown in Figure 3.7, a point, $\boldsymbol{P}$, in the scene is given by,

$$\boldsymbol{P}_{cv} = \mathrm{T}'\,\boldsymbol{P}_{cr} = \mathrm{T}'\,\boldsymbol{P}_{cm} \tag{3.18}$$

where $\boldsymbol{P}_{cv}$, $\boldsymbol{P}_{cr}$ and $\boldsymbol{P}_{cm}$ are the point in the coordinate frames of the virtual camera, the real camera and the mirror respectively. The Y and Z components between the real camera and the mirror coordinate frames are exchanged by direct comparison as shown in Figure 3.7. Equation 3.17 can now be expressed in terms of the mirror coordinate system by,

$$m_x = \frac{x}{f} = \frac{X_c}{Z_c} = \frac{\mathrm{T'}_x\, X_m}{\mathrm{T'}_y\, Y_m - b} \qquad \text{and} \qquad m_y = \frac{y}{f} = \frac{Y_c}{Z_c} = \frac{\mathrm{T'}_z\, Z_m}{\mathrm{T'}_y\, Y_m - b} \tag{3.19}$$

where $\mathrm{T'}_x$, $\mathrm{T'}_y$ and $\mathrm{T'}_z$ are the respective rows of the transformation matrix $\mathrm{T'}$ (ignoring the effects of T_m).

Applying the transformation $\mathrm{T'}$ for the real to virtual camera coordinates yields:

$$m_x = \frac{\cos\phi\, X_m - \sin\phi\, Z_m}{\sin 2\theta\, \sin\phi\, X_m + \cos 2\theta\, Y_m + \sin 2\theta\, \cos\phi\, Z_m - b} \tag{3.20}$$

and

$$m_y = \frac{-\cos 2\theta \sin\phi\, X_m + \sin 2\theta\, Y_m - \cos 2\theta \cos\phi\, Z_m}{\sin 2\theta\, \sin\phi\, X_m + \cos 2\theta\, Y_m + \sin 2\theta\, \cos\phi\, Z_m - b} \tag{3.21}$$

The choice of a “sensible” angle for the mirror plane simplifies the situation considerably. For $\theta = 45°$, Equations 3.20 and 3.21 reduce to,

$$m_x = \frac{\cos\phi\, X_m - \sin\phi\, Z_m}{\sin\phi\, X_m + \cos\phi\, Z_m - b} \tag{3.22}$$

$$m_y = \frac{Y_m}{\sin\phi\, X_m + \cos\phi\, Z_m - b} \tag{3.23}$$

and rearranging yields:

$$b.m_x = \begin{bmatrix} -\cos\phi + m_x \sin\phi & \sin\phi + m_x \cos\phi \end{bmatrix} \begin{bmatrix} X_m \\ Z_m \end{bmatrix} \tag{3.24}$$

$$b.m_y = \begin{bmatrix} m_y \sin\phi & -1 & +m_y \cos\phi \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} \tag{3.25}$$

Equation 3.24 relates a single image measurement x to two unknowns X_m and Z_m and Equation 3.25 relates the y to three unknowns X_m, Y_m and Z_m. Given fixed multiples of ϕ yields multiple instances of Equations 3.24 and 3.25, either set of which can be used to estimate the depth of the imaged scene.

In practice, a set of Equations 3.24 are used to derive depth estimates because of the greater disparity across the image plane from the rotation about the Y_m axis, refer to Figures 3.5 in order to visualize this.

With reference to Figures 3.5 and 3.7 it should be noted that the standard configuration creates a cross-eyed stereo. In this configuration the higher frame number of any two images selected from a periscopic stereo sequence is defined as the right hand image in a stereo pair. This is not however exclusive and a diverging configuration is also possible by exchanging the order of the left and right hand images of the stereo pair. This configuration is not used in any part of this research.

In theory, a selection[3] number of image frames with small[4] displacements between them can be used to improve the accuracy of the estimates of depth and/or position. However, in practice there is a number of limiting factors which must be considered. The following section identifies these factors and also describes the construction of a periscopic stereo head.

3.4 Design, Construction and Operation of a Periscopic Stereo Head

Figure 3.8 shows a sketch of a periscopic stereo head. From the analysis in Section 3.3, the preferred configuration is for a 45° mirror rotating about the camera's optical axis at some known distance along it. While it is feasible to vary the angle of the mirror plane while in use, the added complexity to camera calibration software together with the requirement for accurate measurement of the angle of the mirror plane effectively negates, in practice, any such consideration. The motivation for this research is the reconstruction of remote environments and these may be hostile to precision machinery. The design and construction of the periscopic head is therefore guided by the need for an enclosed,

[3]The scene point must be imaged in all frames so there is a limit to the number of successive frames containing the same point.

[4]The definition of "small" displacement is given in Section 3.4.

robust unit.

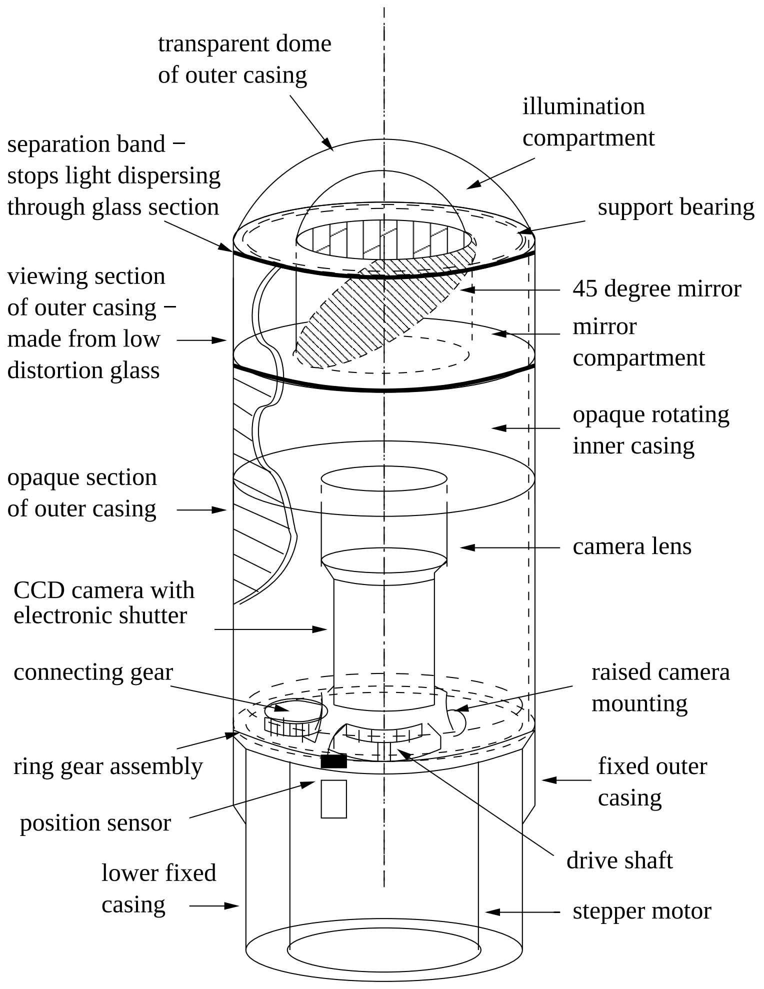

transparent dome of outer casing
illumination compartment
separation band – stops light dispersing through glass section
support bearing
viewing section of outer casing – made from low distortion glass
45 degree mirror
mirror compartment
opaque rotating inner casing
opaque section of outer casing
camera lens
CCD camera with electronic shutter
connecting gear
raised camera mounting
ring gear assembly
fixed outer casing
position sensor
drive shaft
lower fixed casing
stepper motor

Figure 3.8: Sketch of a periscopic stereo head.

The periscopic head is constructed from three tubular sections assembled around a single central axis. The outer casing and lower, inner casing are fixed while the upper, inner section is free to rotate. This upper, inner section is driven by a stepper motor via a ring gear system. The ring gear assembly sits just below the shoulder of a raised section of the lower casing which houses the stepper motor and control electronics. The raised section allows a transmission path from the central drive gear of the stepper motor out to the ring gear, via a connecting gear mounted on the lower section just below the shoulder. The camera, which is chosen to have a reasonably accurate alignment of its optical axis and the centre of the CCD grid[5], is mounted on the raised section and aligned to the central axis. The inner rotating section is opaque and has a plate at the top with an aperture to the mirror compartment. A circular, good quality[6], mirror is mounted at 45° to the central or optical axis. Such mirrors are standard components in small, home, telescopes. The outer casing is opaque, except for a low-distortion glass viewing window and incorporates a transparent dome housing an illumination compartment for use in environments with poor lighting. The wiring for the illumination compartment runs up between the inner section and outer casing and therefore should be kept to the minimum acceptable current rating for the illumination required. The outer casing sits on top of a support bearing which allows the inner rotating section to float freely while offering stability. The lower casing houses the control electronics, consisting of speed control and switching for the stepper motor, via a Hall-effect sensor. The Hall-effect, or scan, sensor also controls synchronization of the image capture buffer which effectively starts and stops each image sequence, or scan, of the surrounding environment.

The distance between the camera and the mirror, together with the angular velocity, effectively determines the baseline distance between the virtual stereo cameras. It might, therefore, be useful to be able to vary the distance between the camera and the mirror. However, this complicates the design and construction of the head. The baseline distance is, at present, fixed according to the viewing angle of the lens such that the edge of the mirror plane is never in view (that is the reflected view of the scene completely fills the camera's field of view). This effectively equates to being proportional to the overall size of the periscopic stereo head. There is no need for the use of a wide-angle, or zoom lens,

[5]The position of the optical axis is assumed to be coincident with the centre of the image plane, however, this is not always the case with low-cost cameras. This is covered in more detail in Chapter 6.

[6]Imperfections in the mirror will translate to the image data.

in this context; a standard camera lens capable of focusing on the mirror is sufficient.

The accuracy of the speed of rotation needs to be “relatively” high so fine pitch stepper motors, operating in micro-step mode, are recommended. In practice, small fluctuations in the motor speed are negligible and are not expected to affect the performance of the image processing algorithms. However, there are obvious limitations to the absolute speed of rotation. With the standard frame rate for video of 25 fps there is a maximum speed of rotation allowable before blurring of the image becomes a problem. This can be offset, to some degree, by the use of a shuttered camera. However image blur should still be taken into consideration.

At the time of writing this dissertation the prototype periscopic stereo head had not be constructed. All experiments were conducted using a shuttered CCD camera (Hitachi KP-M1E/K) suspended above a turntable rotating at 16.66 rpm (16.66 rpm $= 5997.6$ dpm $= 99.96$ dps). Figure 3.9 shows the experimental setup used for creating an image sequence similar to that expected from a real periscopic stereo head. The angular displacement between frames was $99.96/25 = 3.9984$ dpf. With approximately $4°$ between frames the amount of overlap on the imaged scenes allows for a large number of corresponding features to be matched across three, four or even five frames. At this, moderately, low number of revolutions, a shutter speed of 1/125th of second is still necessary in order reduce the angular displacement while the shutter is open to an acceptable $0.032°$ and thereby eliminate any image blur. There is obviously no synchronization between the motor, the shutter and the camera’s frame rate in the “simulated” periscopic stereo head used for the research considered in this dissertation.

A reduction in the speed of rotation in order to reduce image blur, or improve the percentage of corresponding features across a larger number of frames, is not an available option. This is because a reduction in disparity between corresponding image features would lead to reduction of the ultimate accuracy of depth estimation. It is known that the use of more than three frames of image data adds excessive computation for no extra accuracy from the numerical processing for reconstruction [SHB99].

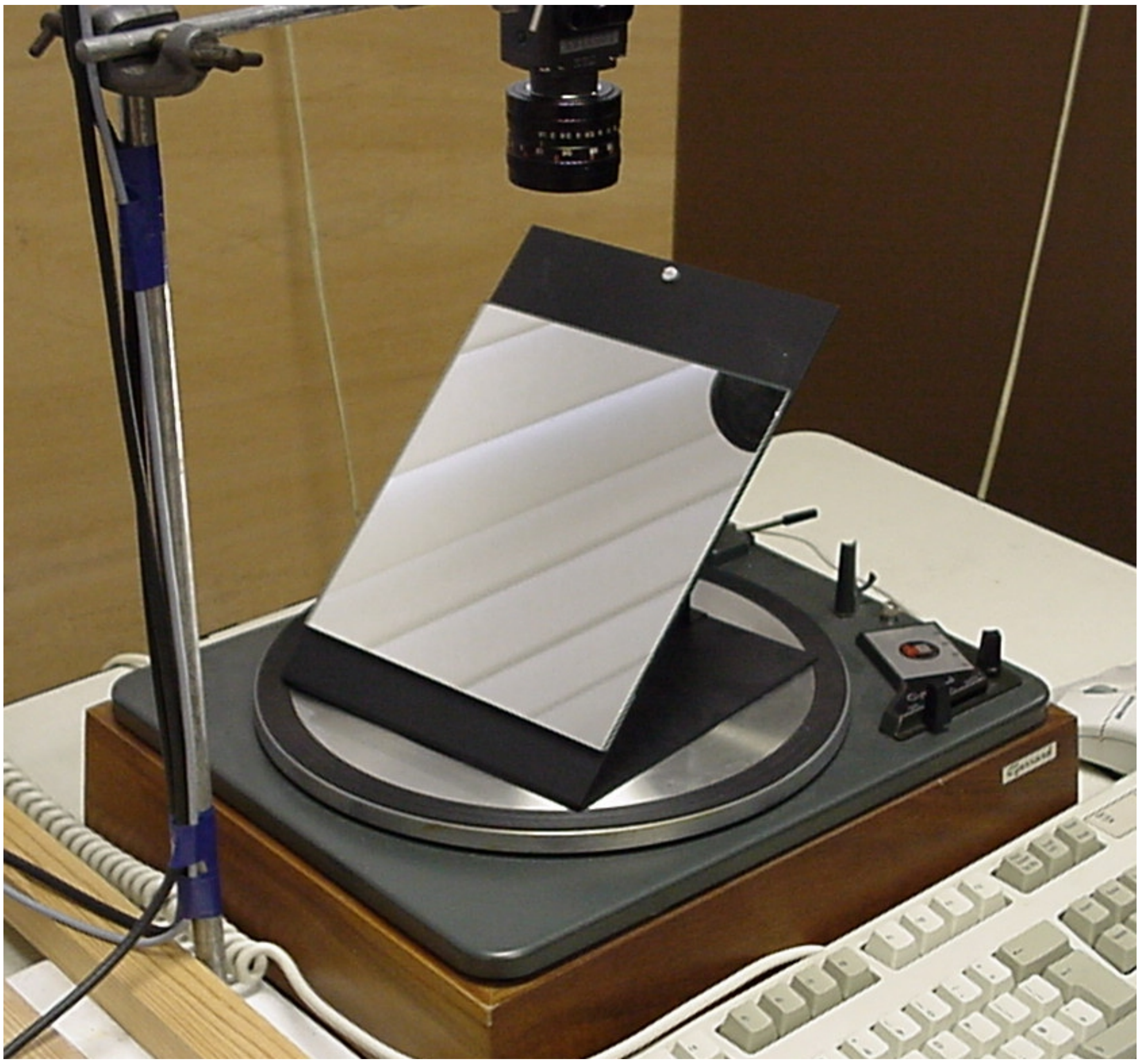


Figure 3.9: Photograph of the experimental apparatus used to implement a periscopic stereo imaging system.

This is discussed further in Chapter 7. A speed of approximately 16 rpm can be generally regarded as an operational minimum. The alternative situation of a small increase in the speed of rotation would allow for better measurements of disparity. However, this would also require the use of faster shutter speeds, which effectively reduces the amount of light entering the camera and subsequently reduces image clarity. Therefore, a practical trade off which must be made. A good initial compromise, determined from experience, is to use only one field[7] per frame and effectively capture only half the image data. An alternative compromise is to use the same rotation speed but select every other frame for processing. Experiments have shown this to produce a usable image sequence.

The video capture card used in all the experiments was a Brooktree Bt848 which supports full 768x576 PAL image resolutions and a number of capture formats. All image streams where captured at 1/2 PAL using BtYUV format and later converted to greyscale PGM images. Capturing 92 (slightly greater than 90 from 4° separation to allow for some overlap at the end of the scan), 1/2 PAL, 32-bit images requires at least 40.7Mb of storage space. Even running on a 200MHz Pentium MMX with 128Mb of RAM the card and the standard software supplied with it is capable of capturing a complete, 360°, scan of the surrounding environment. However, in order to maintain compatibility with a Linux operating system, the use of in-house software was preferred. The Homogeneous Video Capture Interface (HVCI)[8] written by Mike Lincoln [Lin99] streams raw 16-/24-/32-bit RGB, or 8-bit greyscale, video data to a dedicated video file on disk. Using direct access to the card and system RAM, HVCI is capable of streaming large amounts of video data.

[7]Many CCD cameras use two fields, of interlaced pixel elements or rows, per image frame, each of which can be saved to the camera's image buffer via different modes of operation.

[8]Available from: `http://vase.essex.ac.uk/software/hvci/index.html`

3.5 Closing Remarks

This chapter has reviewed the analysis of the relative camera geometry induced by periscopic stereo and introduced, for the first time, a practical design for the construction of a robust periscopic stereo head. The imaging system yields a virtually constant geometric relationship between consecutive image frames. However, it also introduces a tumbling motion to the image data. It has been shown that this tumbling motion can be corrected with standard geometrical transformations which can be applied directly to the image data prior to subsequent processing. However, it has been suggested that the rotational component about the optical axis should be corrected at a later stage.

It should be noted that periscopic stereo is not intended to replace conventional stereo which is predominant for many robotic tasks. Periscopic stereo is not intended to "show you where your going". However, it can record where you have been and it's inclusion in a robot's sensor array allows for increased perception for both robotic systems in, and subsequently human investigation of, remote environments.

In general, this concludes the discussion of the hardware aspects of periscopic stereo. All the remaining chapters in this dissertation deal with the image processing and computer vision software techniques required to realize large-scale scene reconstruction using image sequences captured by a periscopic stereo head. The concept of when the rotational correction should be applied and the effects of delaying the correction continue through Chapters 5, 6 and 7, each of which deals with one of the three, mutually dependent, processes that constitute 3D reconstruction.

Chapter 4

Feature Extraction for 3D Reconstruction

A fundamental prerequisite of the 3D reconstruction process is the initial extraction of relevant features from the 2D image data and their representation for subsequent processing. In order to maximize the completeness of any model of the scene these extracted features need to have different levels of interpretation so that any object in the scene can be described in terms of, say, the distribution of some of its points or the number of visible sides or even the area of its connected surfaces. The problem with standard techniques for feature extraction is that they usually deal with just one-dimensional (edge) or two-dimensional (corner) features and then represent these simply as collection of *points* of interest. Historically, this was acceptable for reconstruction algorithms which used ‘edge’ points to recover 3D structure by matching the *corresponding* points in two or more images of the same scene. However, it has been shown [ST98, MK98] that the correspondence of ‘non point-like image features’ such as lines, curves and planes, with their stronger geometric constraints, allows some relaxation of the constraints applied to 3D reconstruction and improves the overall accuracy of the final model. Feature extraction for the purposes of scene reconstruction is therefore more than just the detection of either edges or corners. Ideally this initial, sometimes referred to as *pre-processing*, stage should consist of a single efficient process that deals with features in a consistent way and represents them in a manner that aids the later stages of the reconstruction process.

In general, feature detection algorithms can be defined in terms of the following list of basic requirements:

1. provide a unique response for each feature element,
2. generate good localization to sub-pixel accuracy,
3. be robust in the presence of noise.

These requirements are quoted in many computer vision texts[1] and have been used by many researchers to design feature, or more specifically, edge detectors. However, there is also another important requirement which is sometimes overlooked and that is the computational load of the algorithm itself. While it may be acceptable in some situations to concentrate on accuracy, 3D reconstruction algorithms are inherently computationally intensive so expending valuable time on an excessively complex pre-processing stage, regardless of the increased accuracy, is counter productive. That is not to say that speed should be regarded as more important than accuracy; simply that there should be a sensible compromise between these requirements. The following section reviews some of the problems with the popular edge and corner detectors and suggests that the recently reported SUSAN algorithm for feature extraction gives the optimal compromise for a 3D reconstruction system.

Comprehensive treatments of feature detection can be found in any of the recommended general texts referenced in Chapter 2. There is also an excellent review by Stephen Smith, available on the Internet via *CVOnline*[2].

[1]probably stem from Abdou and Pratt's, *Quantitative design and evaluation of enhancement/thresholding edge detectors, Proc. IEEE.*

[2]see `http://www.dai.ed.ac.uk/CVonline/feature.htm` or `http://www.fmrib.ox.ac.uk/~steve/review/`

4.1 The Problem with Derivative Based Feature Extraction

A considerable amount of research has been conducted on edge and corner detection. Much of this has concentrated on improving methods which are based on calculating either the first and/or second derivatives of the image intensity function. A variety of alternative methods, ranging from morphology to surface fitting, have also been reported but most are computationally intensive and do not yield any significant improvements over the standard, derivative based methods [Smi97]. In the opinion of the author of this dissertation, the most popular derivative based feature detection algorithms, have been presented by; Canny [Can86] with extensions by Deriche [Der87] or Haralick [Har84] for edges and Kitchen and Rosenfeld [KR82] or Harris and Stephens [HS88] (also referred to as the Plessey algorithm [Nob88]) for detecting corners. It is well known in signal processing that derivatives enhance the noise component as well as the information component of a signal [Ben88]. Their use is therefore, usually accompanied by some form of filtering. The inclusion of filtering in images effectively blurs any features and introduces error in determining their accurate position. Various types of filtering, that have an ideal, *isotropic*, response, most prominent of which being Gaussian, have been employed in order to minimize this positional error. However, these are essentially implemented by discrete kernels which can, at best, only approximate the desired properties. Therefore, even before any other problems are considered, there is an immediate conflict between robust detection and localization accuracy which requires some trade-off.

A second, less obvious, problem is the ability, or rather inability, to produce edge enhanced images with good connections at junctions. This is, in some respects, a continuation of the inclusion of filtering which tends to "round-off" corners but is also as a consequence of the inherent differences between processing one and two dimensional features. The separate development of good edge detection and good corner detection is testimony to the fact that a combined, edge and corner detection algorithm with a high level of performance has not been an easily solvable problem. This problem of "good connections" is often considered as a concern for subsequent edge linking algorithms which are used to form *strings* of edge pixels that can be modelled by either straight lines or curves. However, the accuracy of these strings and the geometric primitives which they form are directly influenced by the performance of initial detection stage. This problem may appear trivial, especially when considering the use of good corner detection algorithms which could "fill in" the gaps left by an edge detector. However, the use of separate processes does not necessarily improve the accuracy or completeness of the subsequent model of the imaged scene, but does incur considerable computational costs. No single edge detection algorithm, that has been developed to date, can yield the perfect response to all edge types or cope with all types of junction. Indeed, it can be argued that most edge detection algorithms are effectively "tailored" to specific edge types and favor the construction of particular edge strings. There is, therefore, a second conflict between extracting the maximum amount of structural information from the image while minimizing the complexity of the algorithm and its computational cost.

Feature extraction for reconstruction, ideally, requires a composite approach that yields both one and two dimensional features and represents them in a way which allows the formation of a range of geometric primitives from simple points and lines to curves and surface patches. These are not new ideas and many attempts have been made to produce an algorithm which produces such composite data. In order to illustrate this a short review of a recent project, which was considered for use in 3D reconstruction by the author of this dissertation, is included. The comments made below are included only to serve as an example of the difficulties of designing a combined feature detection algorithm and not as an in depth critique of this system.

The "FEX" (Feature EXtraction) [För94] project has reportedly developed a versatile feature detector that produces a *feature adjacency graph (FAG)* of points, lines and "blobs" (homogeneous regions) from the initial image in a single stage process. The idea behind this project ("to classify features and centralize the data in a single output image which also incorporates neighborhood relationships") [För94] is of interest to 3D reconstruction since a lot of structural information is encoded. However, the implementation of the key components in this particular project still suffer from some fundamental problems. Firstly the basic feature extraction relies on standard gradient based algorithms, which as discussed, suffer from the contradiction of localization accuracy versus noise suppression. Most of the gradient-based feature extraction algorithms require a parameter to control the scale of filtering used and this is largely dependent on the type of feature to be recovered. In this case, an estimate of the scale required is calculated from some "local" image statistics and then used in a number of rather complex stages of convolution to yield the initial responses. This scale-space approach is well known. However, a new combination of geometric measures is reported [För94] which yield junctions and short, straight line, segments. It is unclear from the paper whether the computationally intensive method employed produces more accurately located features or simply enforces very stringent segmentation to ensure the production of a tri-class or "ternary" image. The paper does, however, admit that junction connectivity in the initial process is poor and that a second process, called FAGANA, is used to improve the segmentation accuracy of the image by bridging the gaps and closing inconsistent line or point structures. The added complexity of classifying each pixel in the image and then producing an accurate (which is, in itself questionable) feature adjacency graph seems computationally excessive when subsequent processes are unlikely to make full use of all this information. In practice, images contain regions which are often difficult or impossible to classify. In the case of 3D reconstruction, systems tend to employ techniques which are conservatively biased to ignore possible errors. In such circumstances producing a greater volume of data which contains a proportional amount of erroneous elements is counter-productive.

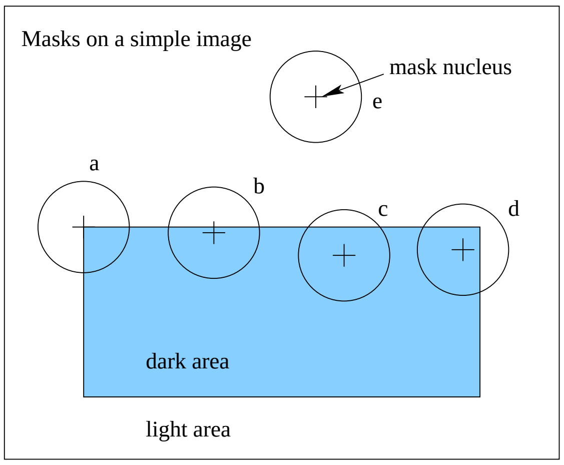
Even though this implementation has some fundamental problems the aims and the basic concept warrant further investigating at some stage in the future. For now, it is concluded that this approach does not, as yet, offer any improvement over other techniques.

Until recently there have been few practical alternatives to derivative-based feature detectors. However, the SUSAN feature detector [SB97], developed by the Robotics Group at Oxford University offers a possible alternative. The following is review of the SUSAN algorithm for feature extraction.

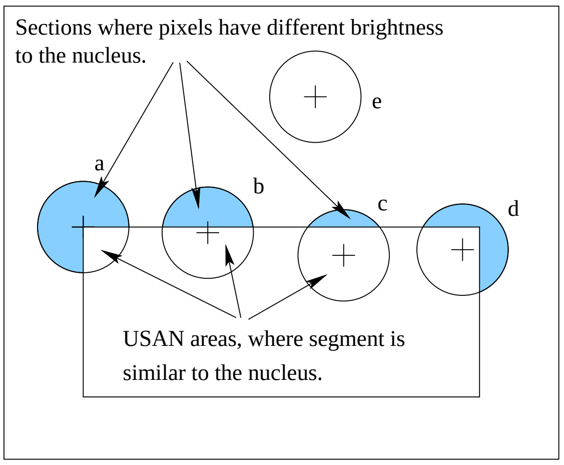
4.2 The SUSAN Algorithm of Feature Extraction

A new approach to feature detection and low-level image processing, referred to as SUSAN, was reported in [SB97]. Its principle of operation is different from many previous feature detection algorithms because it does not attempt to calculate the derivatives of the image data. Instead, an approximately circular mask is used to calculate the local area of *similarity* to the mask's central pixel, referred to as the *nucleus*, across the whole image. This is effectively a form of local integration of the image intensity function. The SUSAN algorithm has, therefore, a certain degree of noise rejection, or filtering, inherently built in. There is little need for any prior filtering stage, which improves localization, connectivity and overall algorithm efficiency. Since the technique for feature detection is based on the ratio of the area, the localization of features is independent of the mask size. The mask size is therefore chosen to maximize the digital approximation to a true isotropic response. This is derived from the analysis of standard Gaussian filtering which gives an optimum response with a mask radius of 3.4, equating to a 37 pixel mask (7x7 mask minus three pixels in each corner, approximating a circular mask) [SB97].

Figure 4.1 has been recreated from the original paper to demonstrate the basic idea. Figure 4.1(a) shows several masks placed on a simple image at particular locations. Figure 4.1(b) indicates which areas are similar (shown as unshaded, or white) and which areas are dissimilar (shown as shaded,



(a) Masks over image


Sections where pixels have different brightness to the nucleus.
e
a
b
c
d
USAN areas, where segment is similar to the nucleus.

(b) Areas of similarity

Figure 4.1: Basic concept of the SUSAN algorithm for feature detection.

or blue) to the mask nucleus. Extracting features from a similarity measure becomes apparent from consideration of the ratio of the shaded to the unshaded areas within each mask in Figure 4.1(b). The mask labelled '`e`' is situated on a homogeneous region and has a maximum area of similarity. When the nucleus of the mask is close to an edge, the area of similarity, referred to as the 'USAN' area, falls to approximately half that of the total mask area. If the nucleus is close to a corner the 'USAN' area falls even further to approximately one quarter of the total mask area, as shown with the mask labelled '`a`'. By subtracting the USAN area from the total area of the mask a measure of the proximity to a feature is achieved. The smaller the area of similarity, the more prominent the indication of a feature. This gives rise to the acronym SUSAN, "Smallest Univalue Segment Assimilating Nucleus".

This simple and effective feature detection algorithm is controlled by two factors. The first "brightness", or similarity, threshold controls the initial response and second, "geometric", threshold determines the selection of either one- or two-dimensional features. Both of these thresholds were derived empirically [SB97], but there is little need to question their validity considering the simple nature of the algorithm. The USAN area (n) for a given pixel is determined from the sum of similar pixels such that;

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \tag{4.1}$$

where, using the original symbols [SB97], $\vec{r_0}$ is the vector defining the position of the nucleus in the image and $\vec{r}$ is the vector defining the position of any other point within the mask. c is the similarity comparison function given by;

$$c\,(\vec{r},\, \vec{r_0}) = e^{-\left(\frac{s}{t}\right)^6} \quad \text{where} \quad s = I(\vec{r}) - I(\vec{r_0} \tag{4.2}$$

and t is the brightness, or similarity, threshold and $I(\vec{r_0})$ and $I(\vec{r})$ are the image intensity values for the nucleus and any other pixel within the mask. This intensity, or brightness, comparison function is shown in Figure 4.2 and was reportedly chosen to give an optimal balance between enhancement and stability [SB97]. The smoothing effect of the exponential is obviously preferable to a box filter

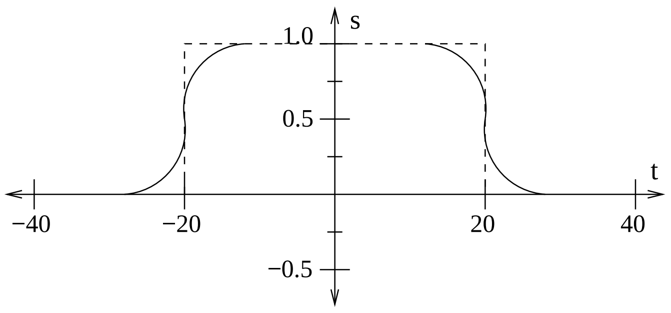


Figure 4.2: Similarity function for SUSAN algorithm.

response and the power to which it is raised can be modified depending on the nature of the image. An extensive justification for this is given in Smith's paper [SB97]. Finally the initial feature response $R(\vec{r_0})$ is determined by comparison with the geometric threshold g such that;

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}) & \text{if } n(\vec{r_0}) < g \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

With reference to Figure 4.1, the value of the geometric threshold for an edge is set to $g = 3n_{max}/4$ and for a corner $g = n_{\text{max}}/2$. An USAN area of greater $3n_{max}/4$ must be a homogeneous region whereas an USAN area of greater than $n_{max}/2$ indicates an edge is near by. From this description of the SUSAN algorithm, it is apparent that the initial feature response will contain multiple indications of the same feature. These multiple indications are eliminated in the usual way by the application of Non-Maximum Suppression (NMS) [Dav97, JKS95, SHB99]. However, there are a number of possible problems in the classification of either one- and two-dimensional features. Smith suggests that these are solved by a few, relatively, simple checks. This is investigated further in Section 4.3. At this point, it is more appropriate to consider edge and corner features separately.

4.2.1 Edge Detection Using SUSAN

Following the production of an initial response, as given by Equations 4.1, 4.2 and 4.3, a second pass through the image is required to further enhance the features and eliminate and multiple indications of the same feature. The application of NMS in the second pass requires knowledge of the feature's direction. Before considering this further it is essential to recognize that the initial processing can produce two distinct types of edge. Smith refers to these as; the "*inter*-pixel edge case", where the nucleus is near an ideal step edge producing only two regions within the mask, and the "*intra*-pixel edge case", where either a thin band or a gently sloping edge produces a small central USAN area surrounded by two dissimilar regions. In the first, inter-pixel, case the edge direction is perpendicular to the vector between the *centre of gravity* (CoG) of the USAN area and the mask nucleus. The CoG of the USAN area is calculated using 1st moments. In the intra-pixel edge case the longest axis of symmetry, calculated from the 2nd moments[3] of the USAN area, is used to directly determine the

[3]The theory of 'moment' calculations is covered in most standard CV texts, see Jain, Kasturi and Schunck [JKS95, chap.2]

edge direction, which is parallel. It should be noted that the use of the term “moments” is misleading since the computation involves summing pixels which have values given by Equation 4.2 and are not binary valued, which is the more recognized use of moment calculations. The use here is therefore more accurately described by the calculation of *volumetric* moments.

The decision as to which case is appropriate for any particular image point is based on the size of the USAN area. According to [SB97],

> “...if the USAN area (in pixels) is smaller than the mask diameter (in pixels) then the intra-pixel case is assumed. ... If the USAN area is larger than this threshold, then the centre of gravity of the USAN is found and used ...according to the inter-pixel edge case. If, however, the centre of gravity is found to lie less than one pixel away from the nucleus then it is likely that the intra-pixel edge case more accurately describes the situation”.

This initial, area-based, condition, described in the first sentence, will be defined herein as *Test-One* in order the differentiate the it from the cases which it determines and the subsequent use of moments.

Little justification, beyond that of “applied” logic, is given in the paper for this important decision. The reason for this, in the opinion of the author of this dissertation, stems from the fact that the mask size is fixed and therefore the ratio of the area of a single pixel wide band to the total area justifies the conclusion in most practical situations. However, the author of this dissertation has found this to be a poorly constructed simplification. The choice for this inter-pixel vs intra-pixel condition does have important implications on the overall performance of the algorithm. This is covered in more detail in Section 4.3.2.

Once the edge direction has been found, using either the 1st or 2nd moment calculations, the initial response is thinned using NMS. The basic algorithm for edge detection using the SUSAN algorithm, simplified and expressed in *pseudo-code* by the author of this dissertation, is shown in Figure 4.3. The original code for all the SUSAN feature detection is available for download from the Oxford site[4]. It

```
Before Any:   for ( range of pixel values )
                  setup similarity look-up table (LUT) for B_thresh;
1st Phase:    for ( all image ) {
                  Mask image to form $M$;
                  assign ptr $M_c$ to LUT based on brightness of mask nucleus;
                  Calc. $usanA = \sum_M M_c - M_{ith}$;
                  if ( $usanA \leq G\_thresh$ )
                     assign init_respn $R = G\_thresh - usanA$;
              }
2nd Phase:    for ( all image ) {
                  if ( $usanA \geq$ MaskWidth ) {
                     Calc. 1st moments of $usanA$;
                     if ($|CoG| \geq 1\times$ pixelwidth ) {
                        Calc. edge direction from CoG_vector;
                        perform NMS across edge;
                     }
                     else jump to '2nd moments';
                  }
                  Calc. '2nd moments' of $usanA$;
                  Calc. edge direction from longest axis;
                  perform NMS across edge;
              }
```

Figure 4.3: Basic pseudo-code algorithm for edge detection using the SUSAN algorithm.

should be noted that Equation 4.2 is implemented with a look-up table (LUT) which is constructed, prior to image processing, for the complete range of pixel values and scaled by 100 to accommodate fast integer operations. Subsequent binary thinning, using Smith's own rules, is suggested [SB97] to eliminate small spurs and ensure *number-of-neighbours* connectivity but this seems to be little different from standard algorithms for the production of edge strings and an unnecessary expense. The use of *hysteresis* thresholds, as used by Canny [Can86] to eliminate the *streaking* effect on edge strings, is not required in any post processing stage [SB97]. The production and use of edge strings is discussed further in Section 4.4 and Chapter 5.

Although not implemented in the original, it is suggested [SB97] that an accurate, sub-pixel, es-

[4]The SUSAN source code can be found at: `http://www.fmrib.ox.ac.uk/~steve/susan/index.html`

timate of the position of the edge can be found by determining the peak of a three-point quadratic curve fit perpendicular to the edge. This sort of extra information, which usually includes edge direction, connectivity and edge strength, or contrast, is often associated with the use of more complex data structures that describe the edge elements (often termed *Edgel's*). Such structures are not implemented in [SB97] but are an essential element of image processing in the `Tina` framework. Modifications to the SUSAN algorithms, discussed in Section 4.3.2, have been derived and developed for use with `Tina`. A small, but important, factor with such data is concerned with the representation of edge direction. In many cases it may be desirable to specify the edge direction in a $0 \rightarrow 360°$ range with some specification for the gradient across the edge (*light over dark* - positive gradients pointing north equate to $0°$). The nature of the SUSAN algorithm does not readily produce this gradient information so the range of orientation is restricted to $180°$.

4.2.2 Corner Detection Using SUSAN

Corner features are processed in a similar manner to that of edges, except that only the larger initial responses are selected for second phase processing by the lower of the two geometric thresholds described on page 60, in Section 4.2 above. Smith suggests that, in practical applications, the use of a wider brightness threshold (≥ 25 in 256 grey-levels) may be desirable to ensure a "suitable" quantity of features. This has not been necessary in the course of the research presented herein and, unless stated, a threshold of 20 (in 256 grey-levels) should be assumed.

In the edge detection algorithm the geometric threshold effectively acts as noise suppression and has no real effect on the selection of one-dimensional features. For corner detection, however, the geometric threshold is fundamental to the selection of two-dimensional features. The indication of a corner is only possible when the USAN area is below half the total mask area and, more specifically, when approaching one quarter. Therefore the smaller the USAN area, the larger the response and the more acute the type of corner. However, some tests are necessary to segment corners from the intra-pixel edge case which are also indicated by small USAN areas, or any other *false positive* indications. The second test, described in the second sentence of the quotation on page 61, determines the absolute distance of the CoG to the mask nucleus by calculating 1st moments of the USAN area. If this distance is small (in this case less than $\sqrt{2}$ of a pixel's width), then the initial response is from an intra-pixel edge case and not a corner. The indicated feature is therefore rejected. An extra test here enforces the contiguity of the USAN area's elements, outward from the nucleus, in the direction of the CoG[5]. According to Smith [SB97], the latter is only really necessary for images with fine, detailed, structures or a high proportion of noise and is included as the final validation of a two-dimensional feature since it does not incur considerable computational cost.

After the tests have validated the presence of a corner, two-dimensional NMS is applied over the response surface, using a 5x5 pixel mask, to remove all but the maximum indication. As with the edge detection algorithm, sub-pixel accuracy can be estimated by fitting a quadratic surface to the initial response, centered on the peak pixel identified by the second phase. This was not implemented in [SB97] but has been introduced in the course of the research presented in this dissertation in order to maintain consistency with the edge detection algorithm and the use of 'Edgel' structures for subsequent use in the stereo matching/correspondence algorithms described in Chapter 5. The feature strength is given by the interpolated value from the quadratic surface fit used to derive the sub-pixel position. Feature orientation is defined as being parallel with the direction of the CoG vector. The pseudo-code algorithm for the modified second phase of the SUSAN corner detection is given in Figure 4.4. In the original implementation, phase two was actually implemented as an extension of phase one with only two passes through the image data being made. However, it is far simpler to consider

[5]This second test is apparent from the fact that the USAN area of a valid corner should be restricted to a small sector within the mask

```
2nd Phase:   for ( all image ) {
               if ( $usanA \leq G\_thresh$ ) {
                 Calc. 1st moments of $usanA$;
                 if( $|CoG| \geq \sqrt{2}$ pixelwidth ) {
                   if ( Contiguous )
                     assign as valid corner;
             } } }
3rd Phase:   for ( all image ) {
               perform 2D NMS;
             }
```

Figure 4.4: Second and third phase, pseudo-code, algorithm for corner detection using SUSAN.

the SUSAN corner detection algorithm in three distinct phases, as shown in Figure 4.4.

4.3 Performance Review of SUSAN Edge Detection

The SUSAN algorithm could be one of the most important low-level, image progressing algorithms to be derived in recent years and yet it has not received the recognition it deserves. This may be due to a lack of faith in its efficiency claims or in the use of some of the conditional thresholds (often referred to as *magic numbers*) in its implementation. The production of accurate, sub-pixel, edge and corner responses is dependent on a few, “relatively simple”, conditional statements which appear logical and consistent. However two of these tests, specifically those in the edge detector, are not implemented with the same degree of precision that is shown in the rest of the formulation of the SUSAN algorithm in [SB97].

From Section 4.2.1, the SUSAN edge detection uses NMS to thin the edge response. This requires knowledge of the edge orientation which is determined using either the 1st or 2nd moments of the USAN area, depending on the which of two cases of edge type, inter-pixel or intra-pixel, occur for a given mask position on, or near, a feature. The decision on which moments to use is based, primarily, on the comparison of the USAN area with the diameter of the mask[6]. This assumes that only a thin

[6]For the 37 cell mask used, the diameter is 7

line of a single pixel wide occurs in practical images. This is an over simplification which introduces the possibility of error. There is no reason why a band of similar pixels, three rows wide, could not occur in a real image. In such a circumstance, the algorithm, as it stands, would conclude an inter-pixel case would exist, even though the calculation of the CoG would lead to a zero length vector to the nucleus and, as such, be unable to determining the edge direction. Smith recognizes this fact in the implementation with a subsequent condition to the inter-pixel case that requires that, if the CoG is found to be within a radius of one pixel from the nucleus, the intra-pixel edge case is assumed. 2nd moments are then calculated and used to determine the edge direction. This is a logical approach to the problem but leads to twice the amount of processing than that required for either of the two cases alone. This occurrence should, therefore, be kept to a minimum. Figure 4.5 shows the fragments of source code (reproduced from the original version, with comments replacing the standard code which is not of interest here) for the two conditional statements which are in question.

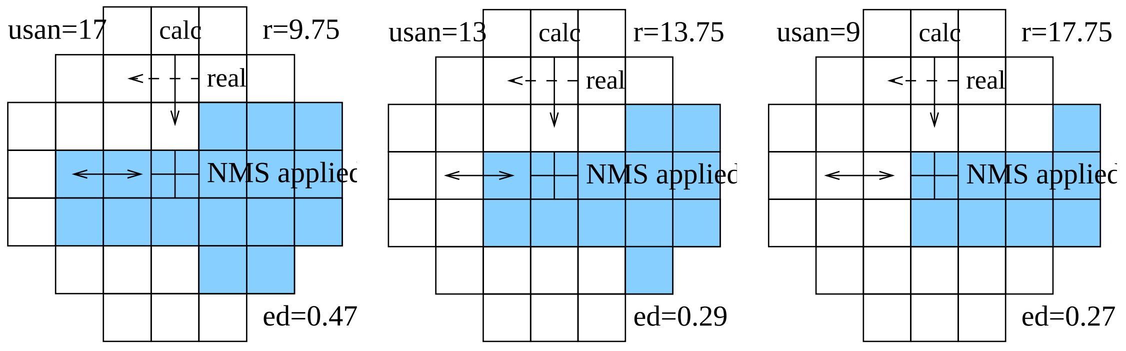
```
........
   if (n>600)
   {
      ..........
      /* Calculate 1st moments. */
      ..........
      z = sqrt((float)((x*x) + (y*y)));
      if (z > (0.9*(float)n)) /* 0.5 */
      {
         do_symmetry=0;
         ..........
         /* Determine edge orientation and carry out */
         /* non-maximal suppression across the edge. */
      }
      else
         do_symmetry = 1;
   }
   else
     do_symmetry = 1;

   if (do_symmetry==1)
   {
      /* Calculate and use 2nd moments */
   }
```

Figure 4.5: Fragment of original source code showing inter-, intra-pixel edge case conditional statements.

The following is a detailed review of three typical inter- and intra-pixel edge cases that demonstrate the concerns with the exact values of the condition variables used in the SUSAN edge detection algorithm.

4.3.1 Analysis of the Threshold Problems in the SUSAN Edge Detection Algorithm

Figures 4.6 and 4.7 illustrate examples of where the conditional statements in second phase of the SUSAN edge detection algorithm, as shown in Figures 4.3 and 4.5, do not always ensure that the correct case is chosen. In the following discussion the values of the USAN area are simplified, such that each pixel adds '1' to the total, and the moments are scaled accordingly. Figures 4.6(a)–4.6(c)



(a) edge deleted (b) edge deleted (c) valid edge

Figure 4.6: Examples of suspect cases in SUSAN edge detection where the mask nucleus is (a) and (b) approaching and (c) at a 30° corner of a large USAN area.

show the mask nucleus approaching a 30° corner of some large area of similarity. In Figure 4.6(c) the USAN area is small, only 9 pixels since the nucleus is over the corner, and the initial response is high, 17.75. The geometric threshold for a corner[7] is 26.75 In Figure 4.6(b) the USAN area is 13 and in Figure 4.6(a) 17 with the initial responses of 13.75 and 9.75 respectively.

Although edge responses are being considered, these particular examples have been chosen specif-

[7]The first phase, or geometric, threshold is given by $0.75 \times 37 - 1 = 26.75$.

ically because they reflect the subtlety of the problem which only manifests itself near corners or junctions, thus leading to increased gaps in edge strings. In Section 4.1, it was stated that one of the problems with gradient based feature detection was that of poor connectivity at junctions. The SUSAN algorithm reports [SB97] to have very good connectivity at junctions. However, this can be either an advantage or disadvantage depending upon how the edge strings are subsequently processed. If simple, "line", strings are to be matched, then the lack of connectivity at junctions would mean that there would be little need for any "long-string" segmentation and therefore much easier approximation, or modelling, of the lines. Alternatively, if homogeneous regions, or *loop* edge strings, are to be extracted, good connectivity simplifies segmentation and reduces the need to bridge any small gaps around the periphery. In either case, the decision on the use of 1st or 2nd moments should not be left to such an "apparently" arbitrary choice of thresholds which are *hard-coded* into these conditional statements.

In Figure 4.6(a) the USAN area is 17 so the inter-pixel edge case is assumed and the CoG of the USAN is calculated. The 1st moments of the CoG are $CoG_x = 15$ and $CoG_y = 7$ respectively and calculating the normalized magnitude of the CoG vector using;

$$|CoG| = \frac{1}{usanA}\sqrt{(CoG_x^2 + CoG_y^2)} \tag{4.4}$$

yields, in this case, comparison figure of 0.973. Equation 4.4 is another way of expressing the second inter-pixel vs intra-pixel conditional statement described on page 61 of Section 4.2 which is defined as $\sqrt{(CoG_x^2 + CoG_y^2)} > 0.9usanA$ in Figure 4.5. The latter will be referred to herein as *Test-Two* in order to differentiate the original conditional statement from Equation 4.4. Note that the original SUSAN threshold of 0.9 allows for the central pixel which is always assumed to be in the USAN area and therefore not directly counted.

A value of 0.973, for the case, above suggests that the edge direction of the example in Figure 4.6(a) can be calculated from the 1st moments as 7/15, or 0.466. This is less than the 0.5 lower condition for the direction of NMS[8] and therefore defined as being vertical. Reference to Figure 4.6(a) shows that this is clearly not the case and the application of NMS across the derived vertical edge would lead the comparison of the features along the central row of the USAN area and the subsequent removal of this particular edge element.

Figure 4.6(b) demonstrates the situation where the mask has moved to the adjacent pixel. At this point the initial response is greater. The shape and therefore the moments of the USAN area have changed to $CoG_x = 17$ and $CoG_y = 5$. Evaluating Equation 4.4 yields a figure of 1.36 suggesting 1st moment calculations for the edge direction, leading to the tangent of 0.29 and another vertical edge being reported. As with Figure 4.6(a), this edge element is removed by incorrect application of NMS along the edge instead of across it. It is only when the mask reaches the corner of this region of similarity does this situation improve. In Figure 4.6(c) the 1st moments are $CoG_x = 15$ and $CoG_y = 4$, yielding $|CoG| = 1.7$ and another vertical edge direction from the tangent of 0.27, this time "arguably" correct[9]. This element is not removed, regardless of edge direction, because it has the maximum local response. This particular case is of further interest. If, for argument, the intra-pixel edge case is assumed and 2nd moments are used to determine edge direction, the element is still kept but the calculation would yield a horizontal edge. This discrepancy in edge orientation is not an immediate problem but does become significant to subsequent edge string processing.

The situation demonstrated by Figure 4.7(a) highlights not only the possibility of incorrect assumptions about the edge direction but also the inefficiency of using both 1st and 2nd moment calculations for a particular edge element. The USAN area is 22 and Test-One, the comparison of the

[8]Edge direction in the inter-pixel case is perpendicular to the CoG vector and the direction for NMS is segmented into; vertical, less than $26.6°$ ($\tan(0.5)$), horizontal, greater than $63.4°$ ($\tan(2.0)$), or diagonal being between these values)

[9]This corner element can be considered as either a vertical or horizontal edge with no subsequent NMS problems, only the feature's orientation is affected

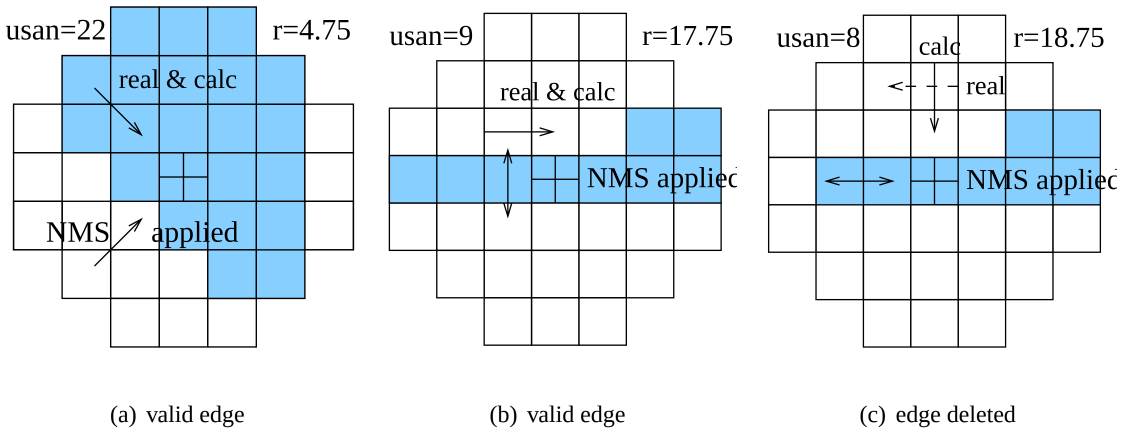


Figure 4.7: Further examples of suspect cases in SUSAN edge detection where the mask nucleus is (a) near a $45°$ corner of a large area and (b) & (c) approaching a sharp $15°$ corner.

USAN area with the diameter on page 61, suggests the use of 1st moments which are $CoG_x = 8$ and $CoG_y = -17$ respectively. However evaluating Equation 4.4 concludes the use of 2nd moments, since the computed value of 0.73 is less than the threshold of 0.9. The use of 2nd moments yields a diagonal edge direction and the correct application of NMS. However, twice as much processing has been carried out to achieve the correct result. At first inspection it may seem obvious to adjust the threshold of Test-One, or even remove it altogether. However, this could lead to greater problems with the more standard vertical and horizontal edges from large USAN areas which could, if using 2nd moments, lead to an incorrect diagonal edge direction. This particular example may not be a problem, as the correct result is produced. However, this case has been highlighted here because of the significance of the computed value of 0.73, which could be used in the second (inter-pixel confirmation) conditional statement. If the threshold in Test-Two was set to say 0.7 then evaluating Equation 4.4 would confirm the the inter-pixel case and the 1st moments used to determine the edge direction. However this would lead to an incorrect application application of NMS and the edge feature being deleted. The importance of this illogical suggestion and the relationship between Test-One and Test-Two will be covered in greater detail in Section 4.3.2.

In the final example, Figures 4.7(c) and 4.7(c) demonstrate the opposite problem of correctly identifying elements in small USAN areas, for instance those created by acute corners. In Figure 4.7(c) the USAN area is 10 and the 1st moments are $CoG_x = 6$ and $CoG_y = -3$ respectively. Test-One would suggest the use of 1st moments since the area is greater than the threshold. However, Test-Two would conclude that the CoG is too close to the nucleus and that 2nd moments should be used. The use of 2nd moments in this case derives a horizontal edge direction and allows the correct application of NMS. Again the correct result has eventually been achieved. However, there would be no reason to carry out the 1st moment, CoG calculations if a slightly higher threshold was applied in Test-One. For the adjacent pixel shown in Figure 4.7(c), the situation becomes much worse. The USAN area is now only 8 and the 1st moments are only $CoG_x = 8$ and $CoG_y = -2$ confirming of the use of 1st moments. However, here an incorrect vertical edge direction is determined with the subsequent removal of a valid element. At the next two pixel positions the edge elements are retained because their initial responses are much greater than the surrounding edge indications. However, the tip of the acute corner has become detached from the rest string.

The above examples have shown that while the SUSAN algorithm is, for the most part, efficient and accurate there are some questions concerning the use of the thresholds in the two inter-pixel, intra-pixel conditional statements. The introduction of a few anomalies may, at first, appear trivial, but their existence leads to the possibility of doubt in the SUSAN algorithm. At best these anomalies lead to reduced efficiency. However, at worse, they lead to incorrect calculation of edge direction, often with a 90° phase shift, resulting in the incorrect application of NMS and subsequent removal of valid edge elements. While these errors amount to no more than a few pixels near corners or junctions they are sufficient to introduce significant errors into the accuracy of features, particularly string elements such as *end points* and *mid-points* which are used in the correspondence of these features and are subsequently projected onto the reconstructed model of the scene. Furthermore the fact that these conditional variables in question can be used to "tune" the final edge response for subsequent processing warrants further study of their use and interaction with each other.

4.3.2 Tuning the SUSAN Edge Detection Algorithm

In Section 4.3 detailed analysis of particular situations where the SUSAN mask nucleus is near corners, or junctions, revealed that, while the basic SUSAN algorithm is sound, the hard-coded implementation of thresholds in the inter-, intra-pixel conditional statements lead to the introduction of errors. Furthermore, the initial inter-, intra-pixel condition should not to be determined by an arbitrary choice such as the area of a single, pixel wide band across the mask. However, the conditional statements could be controlled by an external variable which could improve efficiency and more importantly tune the output. The aim here being to produce either fully connected or distinctly segmented edge strings.

In order to gain a better understanding of the use and interaction of these thresholds the edge detection algorithm was repeatedly applied to the test image supplied with the original SUSAN source code. This is reproduced in Figure 4.8 for convenience.

Each time the experiment was repeated the Test-Two threshold ($|CoG|$) was varied in the range of 0.5 to 1.1 for specific values of USAN area threshold (usanAT) in the range of 600 to 2000 (in steps of 200). Values outside these ranges lead to a rapid degradation of performance, which is logical, considering their function. However, this suggests a motive for the original "hard-coding" of the thresholds. The percentage of pixels which require both 1st and 2nd moment calculations was recorded in each case. Figures 4.9(a), 4.9(b), 4.10(a) and 4.10(b) show the results from these experiments. When the Test-Two threshold ($|CoG|$) is set low, confirmation of the inter-pixel condition is more likely. However, as the threshold rises the condition becomes less likely and the intra-pixel is preferred. The

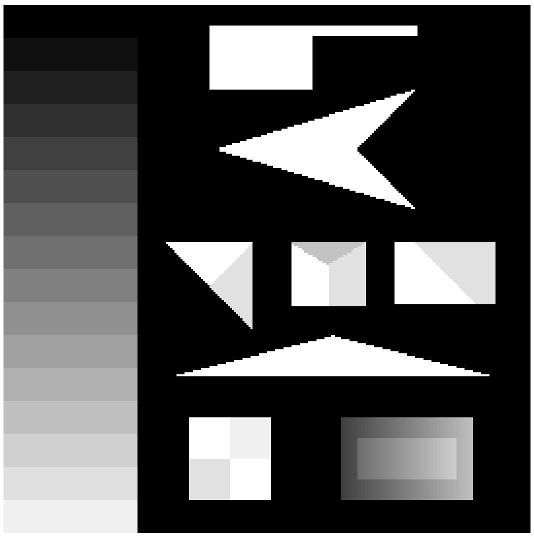
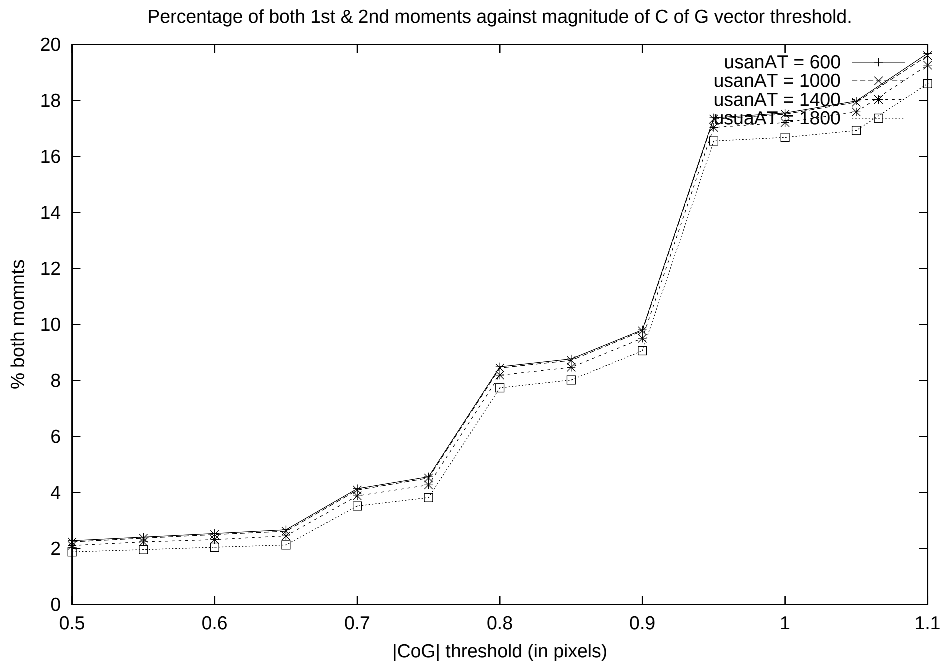


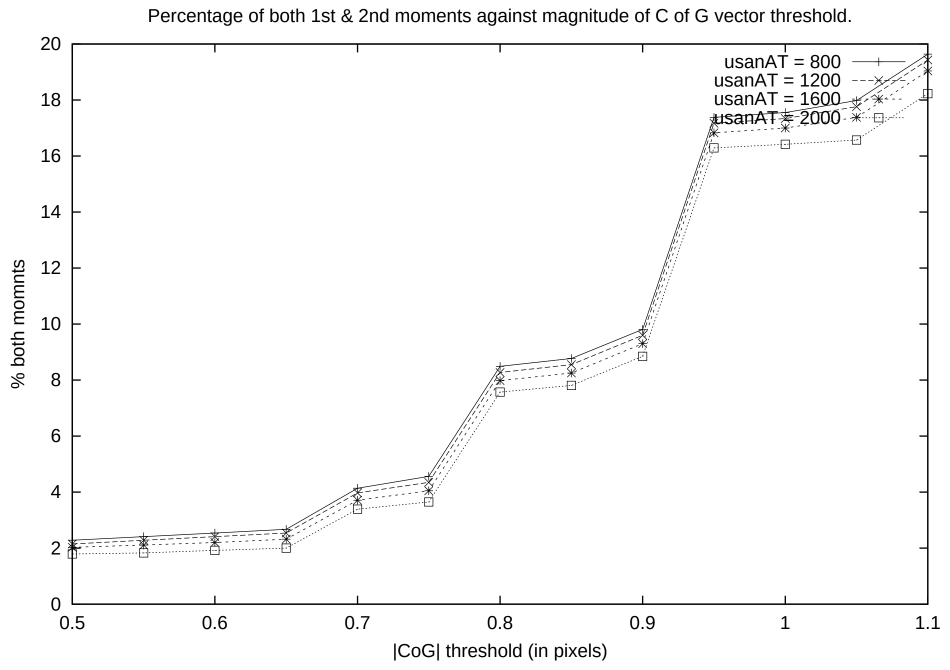
Figure 4.8: Original SUSAN test image.

gain in efficiency is apparent in Figures 4.9(a) and 4.9(b) where the percentage of both 1st and 2nd moment calculations is plotted against $|CoG|$. These results suggest that a threshold above 0.9 leads to a less efficient algorithm. Below 0.65 the algorithm is very efficient and little more can be gained. Varying the USAN area threshold (usanAT) has a small, but noticeable, effect on efficiency. This is logical since the larger this threshold, the greater the number of edge indications that are processed directly by the 2nd moment calculations. However, interestingly, the efficiency gain is not as great as that shown previously. In Figures 4.10(a) and 4.10(b) the effect of varying the area threshold is more apparent. As the area threshold is increased, the number of intra-pixel cases increases. This appears to increase the possibility of error near corners, or junctions, leading to a reduction in the total number of valid edge elements being reported. What is more even interesting, is that this effect is only moderate below 1600 and not significant below a threshold of 1000, where the results are virtually identical.

In Figures 4.9(a) and 4.9(b) each plot follows the same general trend in efficiency so there appears to be little interaction between the two thresholds. However, the results in Figures 4.10(a) and 4.10(b) show that there is some small interaction between the two thresholds. The plots with the higher

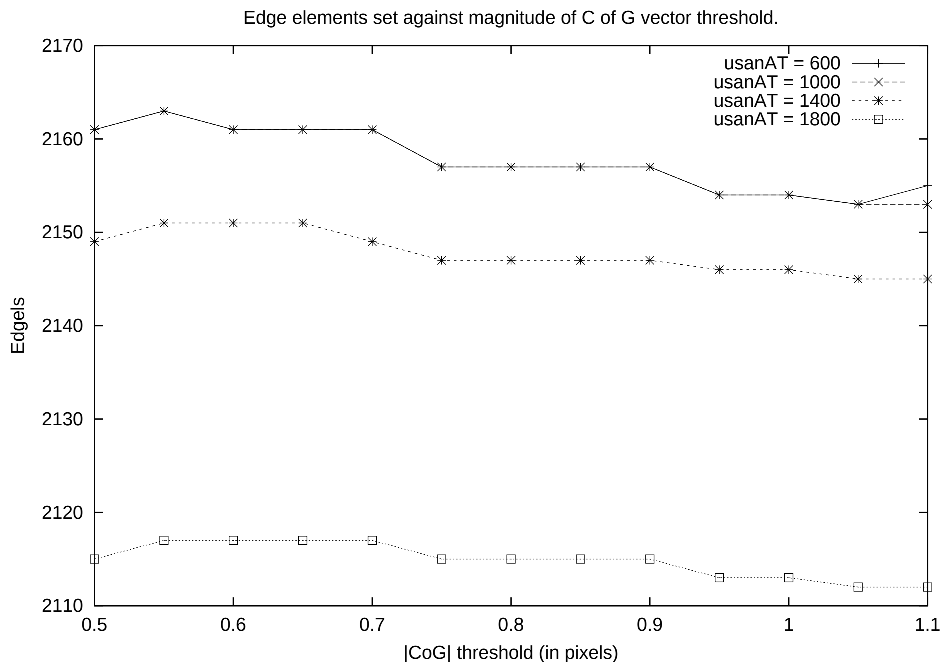


(a) part A: $600 > usanAT > 1800$

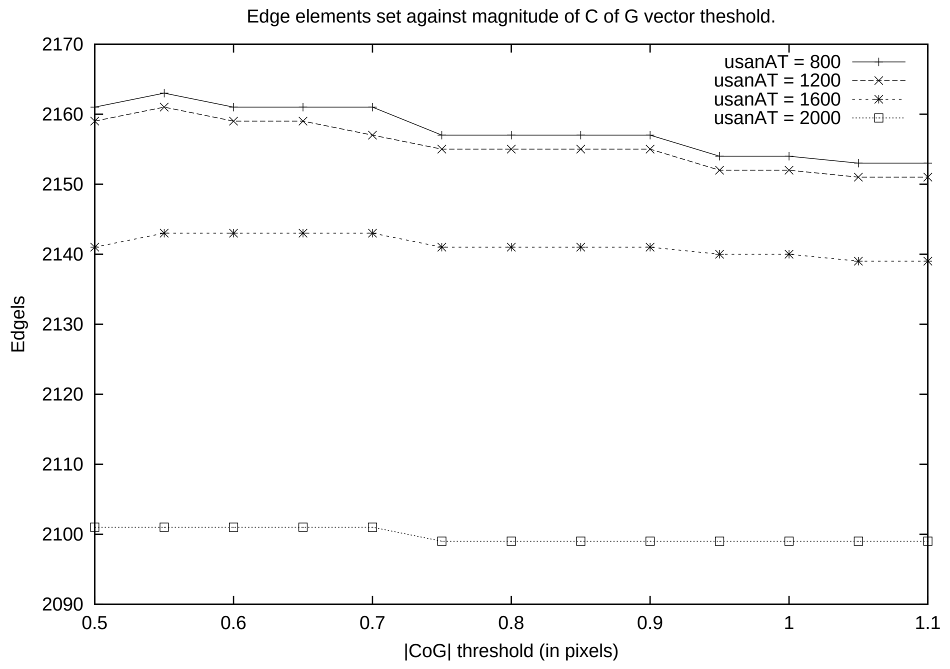


(b) part B: $800 > usanAT > 2000$

Figure 4.9: Results of tuning SUSAN's inter-, intra-pixel conditions – use of moments.



(a) part A: $600 > usanAT > 1800$



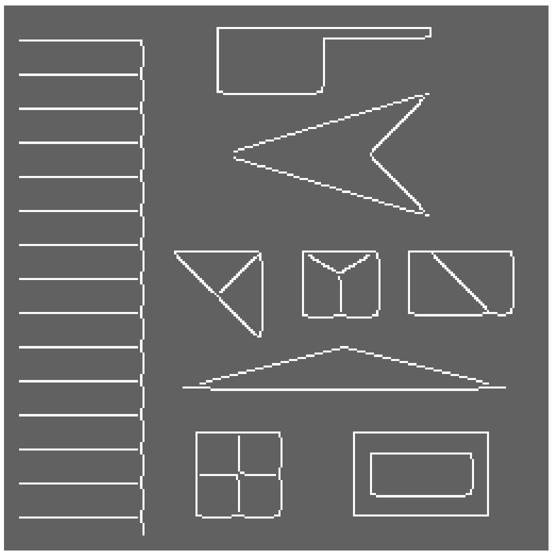
(b) part B: $800 > usanAT > 2000$

Figure 4.10: Results of tuning SUSAN's inter-, intra-pixel conditions – number of valid edge elements produced.

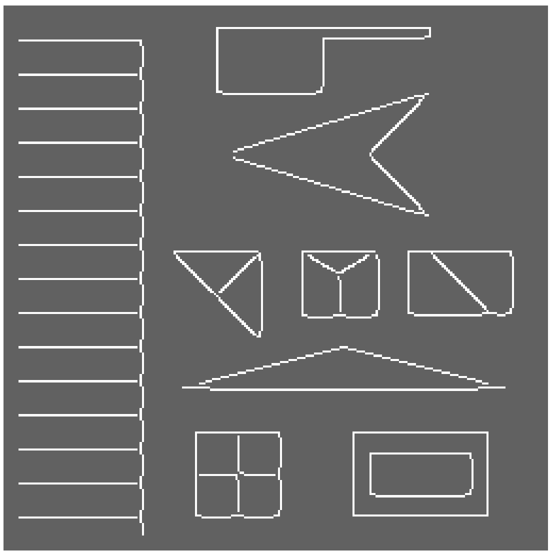
‘usanAT’ thresholds tend to have a flatter response over the range of $|CoG|$ threshold, while the lower ‘usanAT’ thresholds show distinct points of interest at $|CoG| = 0.55$, $|CoG| = 0.7$ and $|CoG| = 0.9$. The maximum number of ‘Edgels’ occurs at $|CoG| = 0.55$ with small reductions at $|CoG| = 0.7$ and $|CoG = 0.9|$. This follows the transitions in levels of efficiency shown in Figures 4.9(a) and 4.9(b). However, this does not reflect the actual quality of the edge response. Although a few extra ‘Edgels’ are displayed for $|CoG| = 0.55$ they appear on the baseline of the shallow isosceles triangle of Figure 4.11(b) (left-hand image, near the corners) and change the single pixel spur shown into a block of three pixels. It is therefore questionable that this is a better quality response than that from $|CoG| = 0.7$. At the $|CoG| = 0.9$, these features are removed altogether. However, the effect on quality can, again, only be regarded as marginal. It is clear from these results that the $|CoG|$ threshold can be fixed at 0.65. Thereby gaining good efficiency, without any significant effect of the quality of the edge response.

Unless an image contains a lot of fine detail, or noise, the majority of edge elements are reported by the inter-pixel edge case from large USAN areas where the use of 1st moment calculations yields the correct application of NMS. This suggests that the primary threshold should be kept relatively low. However the results show that this is not, as essential, as logic would dictate. Higher thresholds produce a gradual reduction in the completeness, but not necessarily quality of the edge response. Figures 4.11(a), 4.11(b), 4.11(c) and 4.11(d)[10] show the edge responses for Test-One thresholds of 1000, 1200, 1400 and 1800, all with the same $|CoG|$, or Test-Two, threshold. For Test-One threshold of 1200 or below, the edge response has complete connectivity around right-angle corners and only small gaps at some ‘T’ junctions and high curvature corners. These small gaps could be subsequently removed with a simple bridging procedure, incorporated as part of an edge string process. For a Test-One threshold of 1400 more small gaps begin to appear at ‘T’ junctions and this trend continues as

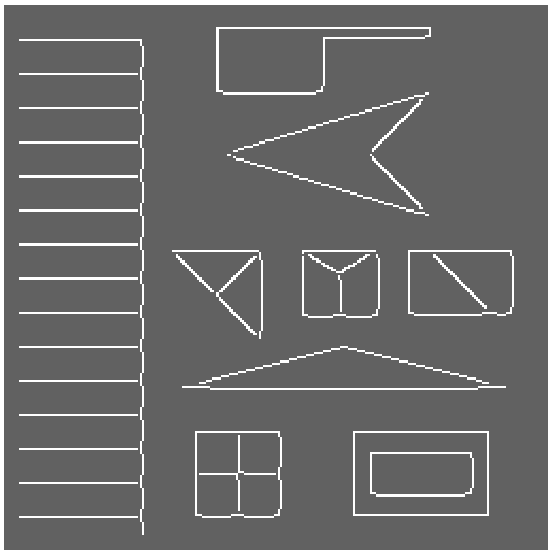
[10]Note: The images are not processed up to the frame limit due to the use of the mask window.



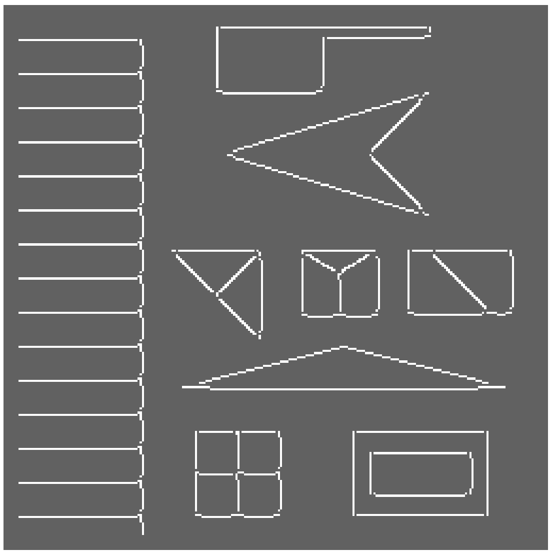
(a) $|CoG| = 0.65$ and $usanAT = 1000$



(b) $|CoG| = 0.65$ and $usanAT = 1200$



(c) $|CoG| = 0.65$ and $usanAT = 1400$



(d) $|CoG| = 0.65$ and $usanAT = 1800$

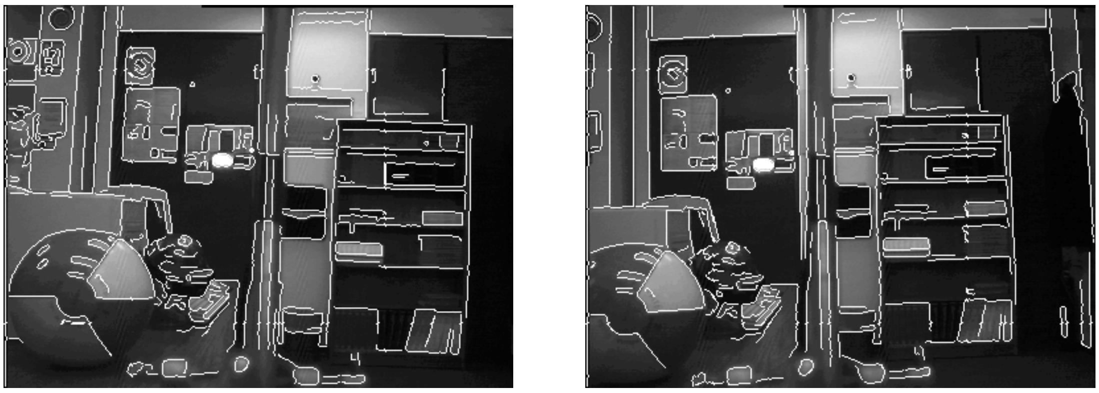
Figure 4.11: Valid edges reported for SUSAN test image demonstrating the algorithm tuning with thresholds at: $|CoG| = 0.65$ and ‘usanAT’ = (a) 1000, (b) 1200, (c) 1400 and (d) 1800.

the threshold increases until, at a threshold of 1800 and above, where no "loop", edge strings exist.

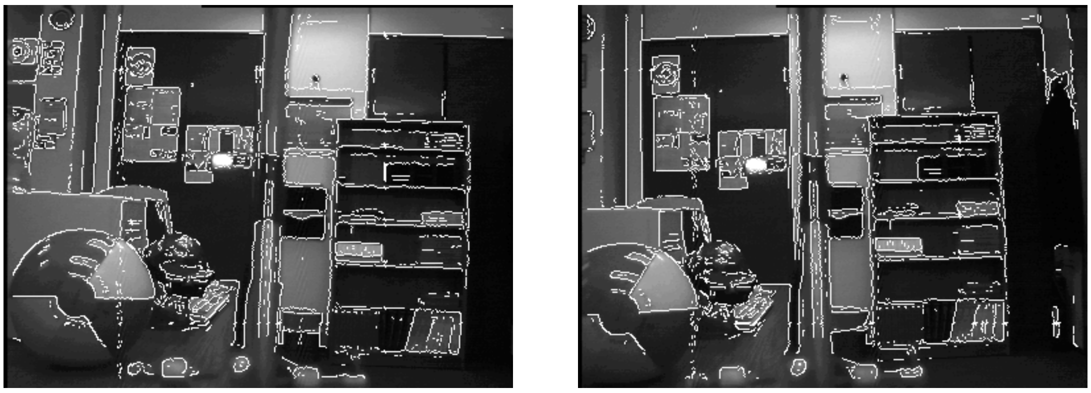
From these results it is clear that the SUSAN edge detection algorithm can be tuned to an optimal efficiency response for both connected and segmented strings, with no extra processing. This is achieved via the application of a fixed threshold in the secondary, inter-pixel confirmation, conditional statement and the use of an external, parameter for the primary, inter-pixel/intra-pixel, conditional statement with the values of either 1200 or 1800 depending on the desired response. Although Smith [SB97] has already conducted a statistical analysis of the SUSAN algorithm, a further statistical analysis should be conducted on the use of the "tuning" parameters introduced in the course of the investigation presented in this dissertation. This should also include a statistical analysis of the sub-pixel localization and feature orientation both edges and corners. A technique for applying such an analysis and modelling sub-pixel accuracy is described in [Roc99]. Such analysis would constitute a considerable distraction from the main emphasis of this research and is therefore recommended for future consideration.

4.4 Integrating SUSAN with TINA

Integrating SUSAN edge and corner detection into the `Tina` framework has highlighted a number of compatibility issues with subsequent processing. The issue concerning the connectivity of edges strings was covered in Sections 4.3 and 4.3.2. Although the use of the extra parameter suggested in Section 4.3.2 assists the generation of edge strings, subsequent processes do not necessarily appreciate this. Figure 4.12 compares the result of string processing the edge images created from both Canny [Can86] and SUSAN edge detection algorithms. Although a similar number of edge elements (approximately 7800) are present in both images, `Tina's` subsequent string processing tends to segment the already connected SUSAN edges, as shown in Figure 4.12(b), while bridging the reportedly



(a) Canny edge strings

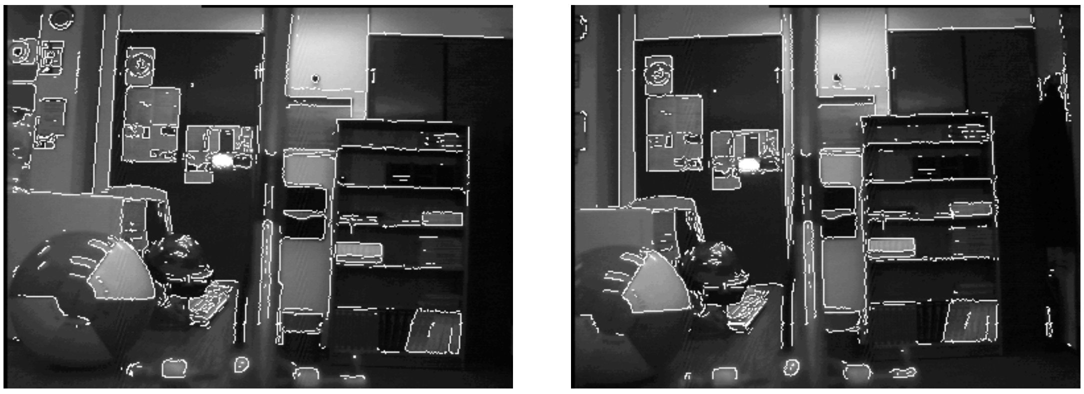


(b) SUSAN edge strings

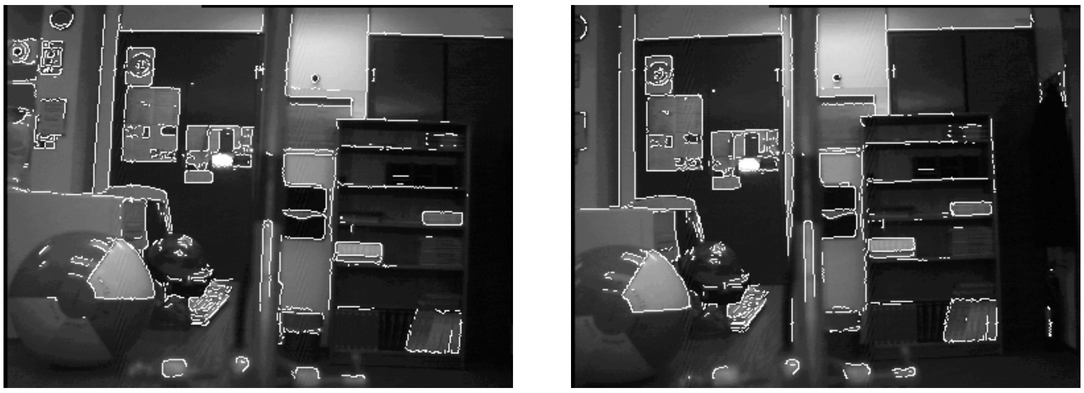
Figure 4.12: Comparing edge strings after (a) Canny and (b) SUSAN edge detection implemented in Tina

less connected [SB97] Canny edges, as shown in Figure 4.12(a). The connectivity of the strings created after the SUSAN edge detection algorithm can be improved by reducing the overall number of edge elements by raising the similarity threshold. In the Figure 4.12(b) was 10 grey-levels (in 256 grey-level) in Figures 4.13(a) and 4.13(b) the thresholds are 15 and 20 respectively. Modifications have been made to the string processing algorithms in `Tina` to switch out unnecessary *hysteresis thresholds*. However, seamless integration has not been achieved to date.

A similar compatibility problem is also apparent in the correspondence matching, see Chapter 5, of



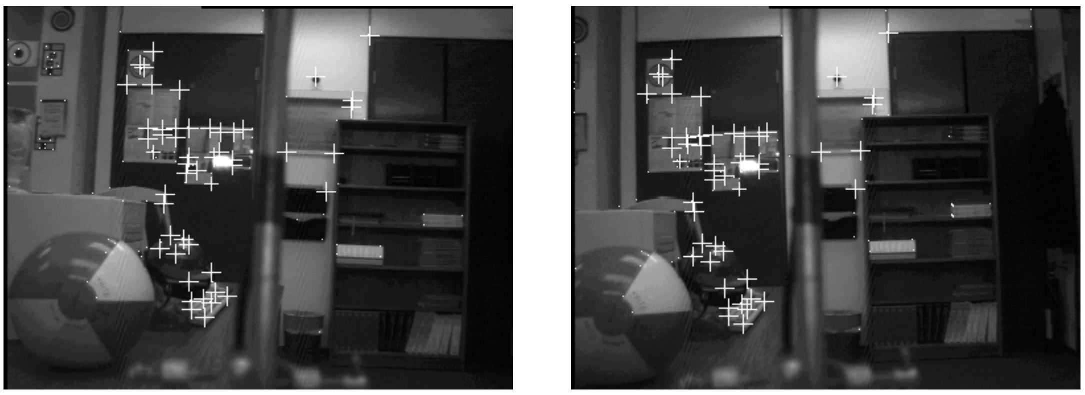
(a) Similarity threshold of 15



(b) Similarity threshold of 20

Figure 4.13: Comparing edge strings after from SUSAN edge detection with the similarity threshold increased to (a) 15 and (b) 20 grey-levels.

corners from both the Harris and Stephens (or Plessey) [HS88] and SUSAN corner detectors demonstrated in Figures 4.14(a) and 4.14(b) respectively. Again, although the number of corners is similar in both cases, over twice as many correspondence matches are derived from the former detector using default parameters. Modifying the similarity and geometric thresholds for the SUSAN corner detector to 15 grey-levels, as shown in Figures 4.15(a) and $0.6\, n_{max}$ (see Sections 4.2 and 4.2.2) improves the situation, as shown in Figure 4.15(b). However, fewer matches are achieved. The correspondence matching is derived from the cross-correlation of the local image intensity function about the feature


(a) matched Plessey corners

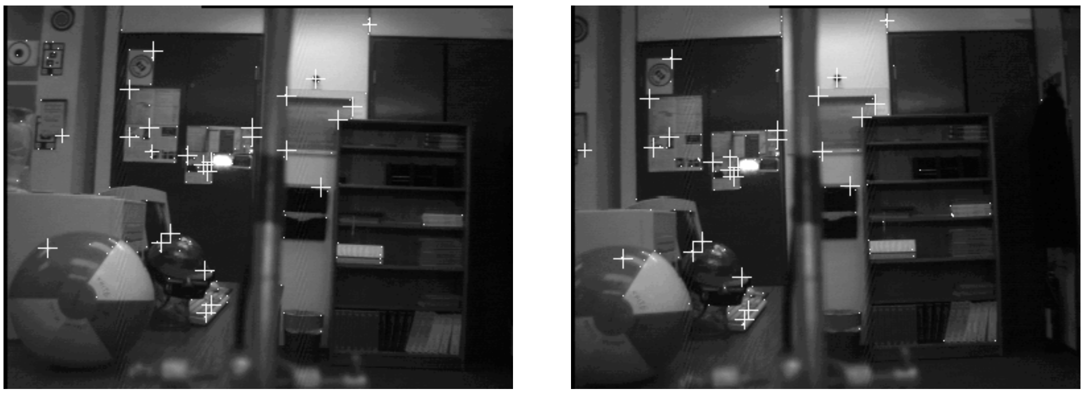
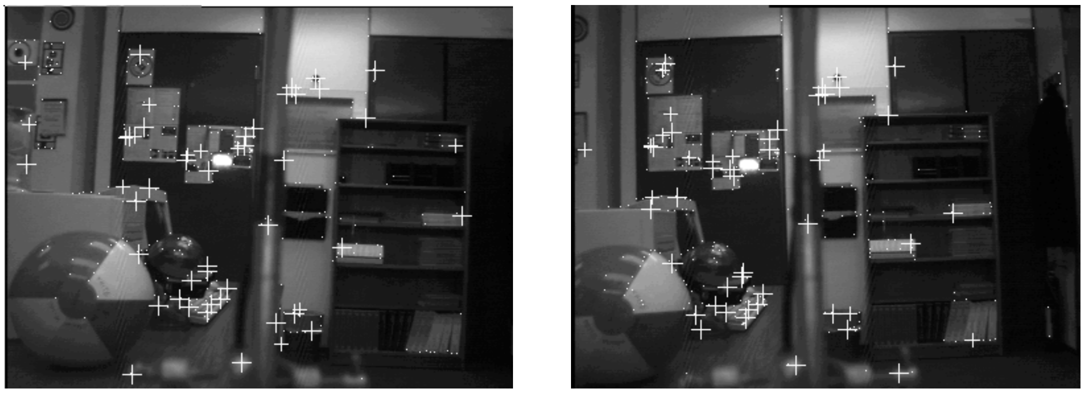

(b) matched SUSAN corners

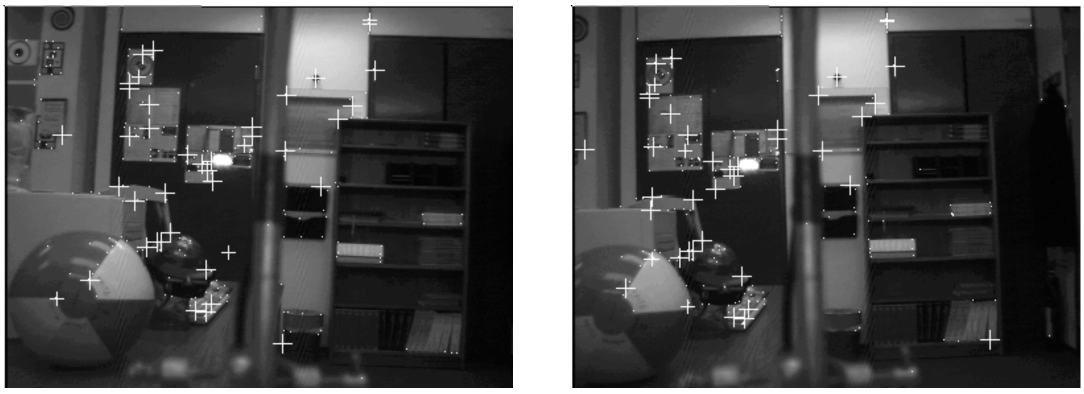
Figure 4.14: Comparing Tina matching after (a) Plessey and (b) SUSAN corner detection algorithms.

position. Examination of the area surrounding unmatched corners reveals that the location of the feature follows the change in profile of the feature and effectively induces a relative shift in the corner location.

Full integration of the SUSAN edge and corner detectors into the `Tina` image processing framework requires further investigation and possible modification to either or both systems.



(a) matched SUSAN corners with thresholds of 15 and $0.6\,n_{max}$



(b) matched SUSAN corners with threshold of 15 and $0.5\,n_{max}$

Figure 4.15: Comparing Tina corner matching after SUSAN corner detection algorithm with modified thresholds.

4.5 Summary

In the opinion of the author of this dissertation, the SUSAN algorithm for low-level image processing is one of the most important techniques to be developed in recent years. It is particularly useful as a pre-process for 3D reconstruction since the basic summing process of the SUSAN algorithm effectively produces a local integration of the image. Therefore, it has an inherent noise rejection capability built into the algorithm and there is no need for any of the extra filtering, which tends to induce localization error in derivative based feature detectors.

The fact that localization of features is reportedly independent of mask size allows for a fixed size to be defined and therefore a simple software implementation. This, coupled with the lack of dependency on any image statistics to derive control parameters for the algorithm, leads to a fast, efficient feature detector. Both edge and corner features can be extracted using the same basic algorithm with only a single test to select the appropriate second stage processing for an accurate, sub-pixel, edge or corner responses. The efficiency and accuracy of the edge response is controlled by two conditions, which although questioned in the course of this investigation, can be used to derive either fully connected or segmented edges for subsequent string processing. However, the use of these requires caution since they are only valid over a limited range. The integration of both SUSAN feature detectors with subsequent processing is not seamless and modification to remove any bias for derivative based feature detectors from these is required for optimal use.

A complete listing of the source code for implementation of SUSAN edge and corner detection, together with modifications detailed in Section 4.3.2 is given in Appendix D. This source code has been written explicitly for incorporation in `Tina` and will therefore not execute in isolation.

Chapter 5

Solving the Correspondence Problem for Periscopic Stereo

Most of the stereo algorithms used in 3D reconstruction require features, identified from differing view points, that correspond to the same real world object, to be *matched*. Matching corresponding features across two or more images continues to be a challenging task in computer vision. In essence it is an optimization problem where the important constraints that can be used to solve the problem are ultimately determined by the solution itself. The features can be either individual pixels or strings of edge elements and the subject, often referred to simply as the *correspondence problem*, has been covered extensively, appearing in almost every text on computer vision. Much of this chapter is, therefore, a discussion of application of known theory. However, few treatments of this subject fully emphasize the fact that while there are many useful techniques that could be applied to all correspondence problems no one particular solution can be applied to *all* problems. In fact, most problems, by the very nature of the features to be matched, the imaging system employed and the subsequent processing for which the match is required, have "uniquely preferable" solutions. This chapter begins with a short review of some of the general techniques and is then followed by a more detailed description of how these have been applied in order to produce sets of corresponding features for use with both calibration and reconstruction tasks associated with periscopic stereo images. In particular these techniques are applied to both rotationally corrected and uncorrected periscopic stereo image pairs, as defined in Section 3.2, in order to assess their compatibility and/or limitations with this image data.

5.1 Constraining the Problem

Techniques for correspondence matching are often grouped into either Feature-based or Area-based methods. This is more of an historic grouping rather than a practical one. At the lowest level of processing this distinction is valid but most modern solutions tend to combine techniques from both. Such classifications can therefore be misleading. In the early days of computer vision research much attention was given to the correlation of images (area-based techniques). The correlation of large areas of an image is computationally intensive so considerable effort was applied to improving performance by tuning the selection of the size and location of the search areas for corresponding points. Area-based methods have the disadvantage that they use the intensity values at each pixel directly and are therefore sensitive to distortions that arise as a result of changes in scene illumination, camera position or the camera's imaging properties. Alternative, feature-based methods, where specific characteristics such as contrast, length and orientation are matched, tended to be less ambiguous. This is due to the lower number of *candidate* matches as well as the lack of dependence on the image intensity function. They also allow higher precision estimates of the disparity between the matched features. However, these also fall short of the desired performance for many real-world systems. Contrary to the impression gained from the considerable amount of early literature on each, the two sets of techniques are not mutually exclusive.

The correspondence problem is inherently ambiguous and there is no such thing as a perfect result. In practice mismatches will always occur in a set of corresponding features. However, their occurrence can be reduced by the application of certain constraints. The following is a description of the constraints which are often applied to solving the correspondence problem. These constraints are derived from either the geometry of the imaging system, the photometric properties of the imaged scene or some specific properties of objects in the real world. Alternative descriptions of these can be found in the text by Klette, Schlüns and Koschan [KSK98, chap.4] or in the text by Sonka, Hlavac and Boyle [SHB99, chap.9].

Epipolar: A pair of points, imaged on separate planes from differing views, that correspond to the same 3D world point are constrained to lie on the epipolar line projected onto each image by the opposing image ray which joins the optical center and the imaged point. This is a geometrical law (described in Appendix A) applicable to stereo camera systems and is the strongest possible constraint which can be applied to the solution of the correspondence problem. It's application effectively reduces the search space from a 2D region to a 1D line but requires knowledge of the relative camera position between the views in order to apply image plane *rectification* which enforces the *canonical* stereo configuration.

Uniqueness: Excluding self-occlusion, where two points which lie along the same image ray are seen as a single point, only one point in the second image should correspond to a point imaged in the first. In practice there are often several candidate matches which could correspond to the original feature. This problem is more apparent when matching isolated point features than edge strings but should always be considered.

Mutual correspondence: This is an extension to the uniqueness constraint and states that correspondence must exist from both left to right and right to left images. This is an obvious requirement of all correspondence algorithms which helps eliminate mismatches.

Photometric compatibility: The image intensities of corresponding image points should be similar. This is a rather weak constraint because it assumes that there is little or no absolute or relative illumination change between image frames which is not generally the case.

Feature compatibility: Only features from the same physical origin should be matched. This sounds the same as the original definition of corresponding image points, however, some extracted image features arise from shadow, or reflection, of objects which are not consistent between views. These type of features should not be used when attempting to solve the correspondence problem. However, isolating these from usable features (arising from abrupt discontinuities on the object surface) is virtually impossible unless some prior knowledge of the imaged objects and the illumination are known.

Geometric similarity: In general, the geometric characteristics of corresponding features do not change a great deal between views. Therefore, a correlation measure between, either, the local regions or some specific feature characteristics should be relatively high. This assumes that the interocular separation is small compared with the scene depth, or that there is little or no rotation about the optical axis between the views. However, if either of these conditions exist then the profile of the imaged feature will undergo some form of distortion and the correlation measure will be reduced.

Disparity smoothness: Assuming a static scene, the disparity calculated for a set of matched features should change slowly across the whole image. This can be visualized by considering two points in the scene, p and q, that are relatively close to each other. Each should yield corresponding image points, in the left and right images, with a small absolute disparity difference. A derivative of this constraint, referred to as the “disparity gradient limit”, is used in the popular PMF correspondence algorithm [PMF85], which is widely referenced.

Disparity limit: There is an absolute limit to disparity, below and above which humans lose the ability to resolve stereo images. In a computer vision sense this translates into the resolution of the camera in one extreme and the amount of image overlap between views in the other. With reference to the geometric similarity constraint above; as the disparity between two views increases the greater the change in the profile of any particular feature and the lower the similarity with the corresponding feature. If there is some prior knowledge of the relative transformation between the two camera views then the disparity across the image should be consistent with the relative camera motion.

Ordering: For an object with a surface of gradually varying depth, two sets of corresponding feature points should lie in the same order along their epipolar lines in both images. This constraint does not apply if there are large relative depths on a single object or between separate objects in the scene.

It should be noted that some of the constraints listed above act in cooperation with each other while others tend to contradict. This is evident when considering particular sets of circumstances such as, the greater the number of corresponding elements in close proximity the higher the possibility of a mismatch due to the increased likelihood of feature similarity. Alternatively, the use of features extracted from regions with high surface discontinuity would allow for more unique matching, yet may tend to be rejected by the techniques that impose geometric similarity, disparity limit and ordering constraints. The use of local properties can achieve reasonable results but global consistency is also required. Ideally the final distribution of matched points should be uniform across the image and reflect the volume of the imaged scene and/or the subsequent reconstruction. In practice this can never be fully achieved. It is clear that the application of these constraints to the particular circumstances of the imaging system, the scene and the subsequent computer vision tasks implies a “tailored” solution.

The following section describes the stereo correspondence method employed for the production of matched image points for use in the re-calibration of periscopic stereo in Chapter 6. The method employed makes use of some of the software tools available in `Tina`, with some modifications. This is then followed by a section which briefly reviews the methods employed in `Tina` for stereo correspondence of both points and strings for subsequent reconstruction.

5.2 Point Correspondence for the Calibration of Periscopic Stereo

In general, correspondence algorithms are designed to process image pairs derived from a camera, or cameras, with particular relative motion between the views. At one extreme the cameras could have a known relative position; at the other extreme, no positional information is available and the views are assumed to be arbitrary. The matching strategy employed must take into consideration the particular geometry of the imaging system used. The more general the relative motion between the views the greater the ambiguity of corresponding match, due to fewer applicable constraints. However, excessive processing to remove ambiguous matches is an unnecessary expense if the relative camera motion is known. This is especially true when considering that subsequent processes should be sufficiently robust to cope with a small percentage of data that are not consistent with the current model of the system, usually referred to as *outliers*. The image pairs, derived from sequential frames, in the periscopic stereo system have some “known” relative motion and therefore allow for the possibility of a simplified correspondence algorithm. The use of the term “known” here is deliberately vague because it is not intended to infer the use of full stereo calibration data which is the more “standard” use of the term “known” when applied to camera motion. The calibration process, for which the set of matched points is required, is not the initial calibration of the system prior to use, but instead, the periodic re-calibration of the imaging while in use. Although the camera system is initially calibrated, the calibration accuracy is reduced over time due to fluctuations in the relative camera geometry. This is more apparent in the case where the rotational correction, identified in Section 3.2, has not been applied. The degradation of the calibration over time is covered in Chapter 6. In general the following applies to both rotationally corrected and uncorrected image pairs but concentrates on the latter.

The "known" relative motion between views allows for the possibility of applying the epipolar constraint, via image plane rectification, and a disparity limit constraint. Due to the fact that the same camera captures both images of the stereo pair and that the time difference between frames is small, both the photometric and the geometric constraints are applied. However, these are implemented in a relatively "loose" manner. While there is rotation of the image data, about the optical axis, between successive views of an uncorrected image sequence, this is small and known. The application of mutual correspondence and the uniqueness constraint are generally regarded as common-place and are included in the algorithms described in this section. Because the correspondence data is required for a subsequent re-calibration process corner edge elements are selected for matching. Camera calibration using image points is discussed in Chapter 6. Since the number of point features in an image is likely to be comparatively low, the disparity gradient constraint is ignored. This is because its use tends to lead to a less reliable solution if the feature distribution is not reasonably uniform across the whole image [TM91]. No assumptions are made about the structure or illumination of the scene so the ordering constraint can not be applied.

Prior to stereo matching, corner features are extracted from both the left and right images and stored as edge elements, or *Edgel's*. In `Tina`, edge, or feature, images consist only of pointers to the features where they exist. The corner locations are identified to sub-pixel accuracy, using local maxima from a 2D quadratic surface fit to the corner strength, or contrast response, and stored along with the peak value of the response and the corner orientation. The specific use of these characteristics will be covered later in this chapter. The ease with which matching techniques can be implemented is directly related to the definition and structure of the feature representation. The definition used in `Tina` is particularly useful because it incorporates a *properties* list which can be used to dynamically store extra data associated with the feature.

The application of the epipolar constraint is possible for image pairs produced by periscopic stereo since an estimate of the relative camera geometry between the views is immediately available. However, in practice, the application of the epipolar constraint involves image plane rectification. This is similar to the rotational correction that can be applied to the image data in the initial pre-processing stage described in Section 3.2 and demonstrated in Figure 3.5. However, a 2D transformation of the image data about the optical axis can not achieve the canonical stereo camera configuration if the images are not originally coplanar. Only the application of a 3D transformation to a common reference frame can yield the desired result.

The standard image rectification process [JKS95] involves interpolating the image data to form a new image plane in the canonical configuration. Such interpolation of the image data is a degradation that tends to reduce the performance of all subsequent processes and should be avoided [Moh93].

A method of simulating image plane rectification, without modifying the initial image data, is given in `Tina`. This is achieved by using a rough, initial estimate of the camera calibration parameters and calculating a rectification matrix for each camera. These matrices are contained in a versatile structure which models a stereo camera system with left and right cameras for both the actual and canonical camera configurations. The matrices therefore map the transformation between the rectified and original image planes for each camera. The source code definition of `Tina`'s camera models is given in Appendix E for ease of reference here and more specifically for detailed use in Chapter 6.

Simulated image plane rectification is implemented by applying the rectification matrix to the position of every feature identified in the original image. The rectified position for the feature is then stored, together with the original, image plane, position, in the ‘Edgel’s properties list[1]. A conditional statement in the access function for feature location selects either the rectified or true image position. The use of access to ‘Edgel’ structures via their address, stored in the image array, is extended in `Tina` to create a *rectified, row-indexed* linked list of the features. This facilitates sequential raster access via an efficient form of look-up-table.

Using the software tools identified above, the search for candidate matches in the second image is simplified (computationally speaking). However, it is not possible in practice to reduce the search space to a single line. The search can be constrained to a band about the epipolar line but no more. Even if the system were fully calibrated, the concept of a perfect epipolar line is not practical because higher accuracy calculations of a feature’s position on the line could lead to many valid matches being ignored. At best, the epipolar band should include ± 1 single row of pixels (simulated via the row-indexed linked list) to allow for the maximum possibility of finding the correct match. While the epipolar band can be widened to accommodate a less accurate estimate for the initial calibration parameters, an excessively wide band would negate all the advantage of applying the epipolar constraint in the first place. This idea of a “variable width”, epipolar band is useful for a dynamic system which automatically maintains its own calibration, but would require a heuristic measure for the initial setting. The default width for the epipolar band, recommended in `Tina`, is ± 3 rows.

Although the search space has been constrained it is still necessary to select the most likely match from a set of possible candidates that lie within the epipolar band. A weak measure of correspondence

[1]Apart from the other useful tools contained within `Tina`, the simulation of image plane rectification using the rectified position stored in the properties list and it’s fast access via the row index linked list was fundamental in it’s choice as the preferred image processing framework to support this research.

is given by the absolute contrast, or strength, values for each feature,

$$\frac{|I_l - I_r|}{(I_l + I_r)} \tag{5.1}$$

where I_i is the intensity of the pixels for the potential match between images the left I_l and right I_r. This is not sufficiently accurate to ensure a low percentage of mismatches. However, it is useful for the rejection of very dissimilar features prior to the next stage of processing (say reject all ≥ 0.6). A more stringent measure is necessary. The standard technique is to apply local area cross-correlation to patches of the original image data centered on the location of the candidate match. The form used in `Tina`[2] is given by:

$$cm(p,q) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} I_l\left[p_x + i', p_y + j'\right] I_r\left[q_x + i', q_y + j'\right] e^{-\left(\frac{i\sigma^2 + j\sigma^2}{4\sigma^2}\right)}}{\sqrt{n\left(I_k\right)^2 + n\left(I_{k+1}\right)^2}}$$
$$\text{where} \quad n\left(I_k\right) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} I_k\left[k_x + i', k_y + j'\right] e^{-\left(\frac{i\sigma^2 + j\sigma^2}{4\sigma^2}\right)} \quad \text{for the} \quad k^{th} \quad \text{image} \tag{5.2}$$

and where p and q are the pixels of the candidate match with coordinates given by $(p_x,\ p_y)$ and $(q_x,\ q_y)$, i and j are the indexing coordinates in the correlation patch and i' and j' are modified coordinates to interpolated values for the image data covered by the patch, $m = n = 2$ for a 5×5 correlation patch, and the nominal value for σ is 3. This is explained later. If the correlation measure (cm) is high (greater than 0.98) then a match is fixed and added to an ordered list of all potentially good matches in the epipolar band for that feature point. These are then stored in the properties list for the feature concerned. Support measures could be added to the correlation measure at this point in order to strengthen the match. However, this is not implemented in `Tina`. In practice the list should contain only a few potentially good matches. A large number of matches would indicate that the threshold for the correlation measure is set too low. The whole process is repeated for every feature point identified

[2]The form of local area correlation use in `Tina` differs from the standard found in the literature by not normalizing to the local image intensity function

in both the left and the right images. The final stage of the whole correspondence process is then to select the best match in each case. This can be achieved using a measure of the match strength already stored or, alternatively, a rejection of all the "less than desirable" matches. The latter is often referred to as *relaxation* and covers a host of techniques that apply additional support measures. Examples of these are the ordering constraint, disparity gradient limit or some combination of global measure that reduces the number of competing matches until the only best remains.

5.2.1 Support Measures for Correspondence Matching

The above are largely standard techniques and examples can be found in many of referenced texts. Of particular interest however, is work published by Thacker and Mayhew [TM91] and Zhang *et al* [ZDFL95]. The first, because of its connection with the software tools described above and the second because it offers an alternative approach which includes an example of a relaxation technique. In the latter, the assumption is for cameras in arbitrary positions and the task is to recover the unknown epipolar geometry. In such circumstances the correspondence algorithm must be particularly robust since the search area consists of large areas of the image and the number of potentially good matches is high. Initially, a standard correlation measure is used and lists of potential matches are constructed similar to that described above. Then a support measure is calculated for all the matches based on a number of constraints. These include a disparity limit and smoothness, both weighted, the uniqueness constraint and a directional compensation. The latter is reportedly [ZDFL95] required because the support measure is not symmetric from left to right and right to left. This method is similar to that applied in the PMF algorithm [PMF85] previously referenced. However, instead of employing a *winner-take-all* or *looser-take-nothing* strategy for resolving the ambiguity, a "*some-winners-take-all*" technique is applied. This correspondence algorithm is reportedly very robust but considerably more complex than more standard approaches and far more complex than that required for periscopic stereo. In the method reported by Thacker and Mayhew [TM91] the correlation measure includes a Gaussian weighting function which allows the measure to vary between 0 and 1 (1 being good)[3]. The assumption made [TM91] is that there is little or no rotation about the optical axis between the two camera views. This does not apply to periscopic stereo as demonstrated in Figure 3.5 earlier.

Delaying the application of the rotational correction of the image data derived from periscopic stereo has distinct implications to the implementation of the local area correlation. If the correction is not applied prior to this stage of the processing chain then the correlation of image patches must be able to compensate for the distortion in the image data. Ideally what is required is a method of local area correlation which is rotationally invariant. The standard, spatial domain, technique for correlation between image planes does not handle rotations of the image data. However, techniques do exist in the spatial-frequency domain [CP76, AR84]. However, these are considerably more complex than is desirable for this particular task.

Techniques that, while not rotationally invariant, are reportedly able to handle large distortions in the feature profile in the spatial domain were proposed by Lane, Thacker and Seed [LTM94]. The techniques, referred to as *stretch*- and *shear*-correlation, apply correlation to *warped* image blocks in a recursive manner until the best fit in accordance with the epipolar geometry is achieved. However, a good estimate of calibration is required for the initial image rectification defined by the epipolar geometry. This is not necessarily available, depending on the level of degradation of the calibration accuracy for which the correspondence data is required. Error in the calibration accuracy is effectively passed to the rectification of the image plane and therefore the localization accuracy of the features. Since these are matched and passed to a re-calibration process a feedback loop is created which can lead to either positive of negative effects. These algorithms also incorporate the disparity gradient constraint, in order to reduce the possibility of mismatches to a minimum, and therefore incur higher

[3]More standard forms (there are a number of alternatives) of correlation give a measure between ± 1

computational costs. These algorithms, unmodified, are not optimal for correspondence matching for subsequent re-calibration considered in this section. They are however revisited in Section 5.4.

An alternative, approximation to the concept of warped image patches can be found in the source code for `Tina`. This method is not referenced and does not appear to have been published. The idea appears (without reference this is an assumption on the part of the author of this dissertation) to attempt to simulate *shear*-correlation by using the orientation of the corner feature to guide the warping of the image patches. This idea has some merit since, in general, the profile change of a feature between views is proportional to the change in orientation. Another alternative approximation to shear-correlation, offered by the author of this dissertation, combines the idea of guiding the amount of warp to apply to the interpolated images patches with the *pseudo*[4] calibration provided by the inherent camera geometry of periscopic stereo. Using the image position of a candidate match as the centre, a warped (rotated about the centre position) 5x5 patch is constructed from the image plane by applying quadratic, surface fit, interpolation to positions determined by either a $\pm 4^\circ$ rotation dependent on a either a left-to-right or right-to-left match, as shown in Figure 5.1.

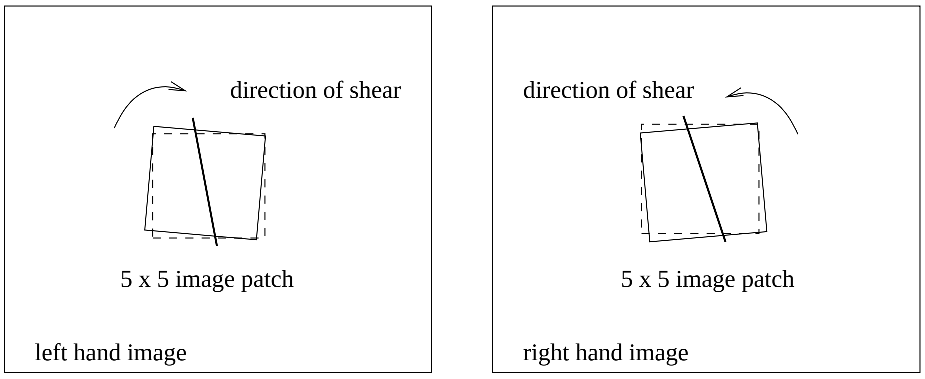


Figure 5.1: Simulated shear correlation with fixed rotation.

[4]The relative camera geometry can be calculated, as described in Chapter 3, and provides a good initial estimate of the epipolar geometry.

Apart from these two ideas of extracting warped correlation patches, a third, more obvious, option is apparent. Due to the fact that the vergence angle between frames is small and the frame capture rate is almost constant, as discussed in Chapter 3, warping the image patches for correlation may not be necessary for an acceptable percentage of mismatches. The number of acceptable mismatches, as discussed in Chapter 6, is open to debate. Not attempting to model the change in feature profile is a questionable premise. In summary, Figure 5.2 shows the pseudo-code for the point correspondence algorithm discussed in this section. The algorithm applies equally to for all three correlation patch techniques, however, the rotated patch is quoted.

```
initially:      from a set of corner Features ($F_{\mathbf{m}_i}$) for $\mathbf{m}_j$ images,
                where $j = 1, 2$.
preproc:        for ( $\mathbf{m}_j$ ) {
                    Simulate image-plane rectification (by constructing
                    a row indexed LUT of pointers to $F_i$)
                    from the "known" relative disparate views;  }
correspondence
matching:       for ( $\mathbf{m}_1$ to $\mathbf{m}_2$ ) {
                    for ( $F_i$ in $\mathbf{m}_j$ LUT ) {
                        for ( epipolar band in $\mathbf{m}_{j+1}$ LUT ) {
                            if ( ! norm($|I_1 - I_2|$) $\geq 0.6$ ) {
                              Compute Correl_measure over 5x5 patches
                              with 2nd image patch rotated by $\pm 4°$;
                              if( Correl_measure $\geq 0.98$ ) {
                                Fix match and store in list with
                                Correl_measure;
                }   }   }   } }
                for ( $\mathbf{m}_2$ to $\mathbf{m}_1$ ) {   Repeat above;   }
                for ( all match lists ) {
                    if ( ! mutual correspondence   &&   unique )
                      throw away match;
                }
```

Figure 5.2: Pseudo-code for the point correspondence algorithm

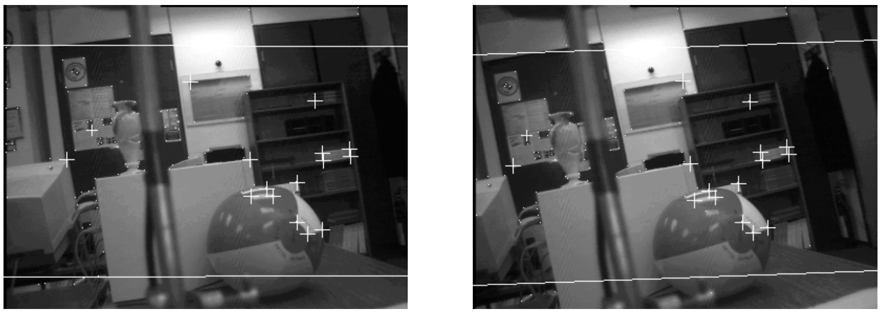
5.3 Assessment of Image Patch Correlation Techniques

Although it is not the aim of this chapter to compare and contrast the performance of image patch correlation techniques, the results presented here were acquired as consequence of testing the proposed algorithms. This is not intended as proof to support either method and the author of this dissertation admits that the final choice of method in this implementation is largely intuitive.

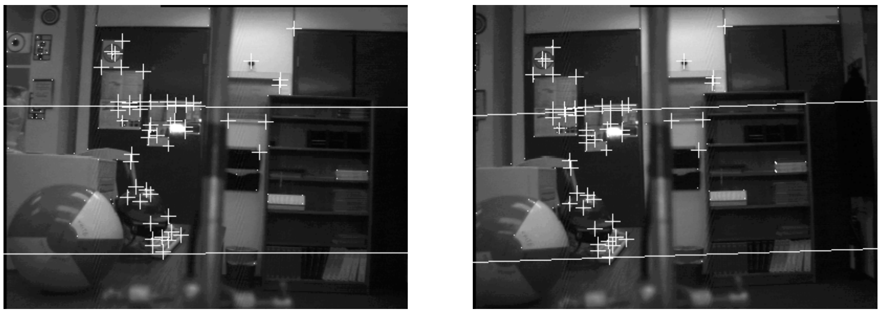
The performance of correspondence algorithms is largely subjective and there is little evidence in the literature of statistical analysis. A notable exception was presented by Thacker and Courtney [TC92]. However, such extensive comparative methods are beyond the scope of the research presented in this dissertation. The following performance measure is only used to give an indication of performance.

$$P_M(m_1, m_2) = \frac{(M_t - M_b)}{F_{min}} \tag{5.3}$$

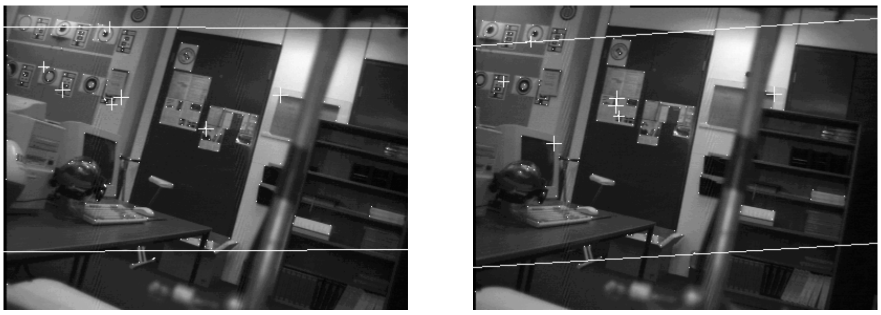
where $P_M(m_1, m_2)$ is the performance measure for a given stereo image pair, M_t is the total number of matches (*Fixed* plus *Good*, or $M_f + M_g$), M_b is the number of bad matches. Fixed matches are those computed to be the most preferred according to the correlation measure and good matches are those in the list of candidate matches which fall inside the uniqueness constraint. F_{min} is the lower of total number of features in either the left or right-hand images. The maximum possible score is 1.0 since if all the possible matches were good, with no bad matches, this would equal F_{min}. The number of bad matches was derived by inspection of the stereo pairs, examples of which are given in Figure 5.3, using a manual , "matched feature", selection tool available in `Tina`. Arbitrary epipolar lines are shown in Figures 5.3(a) and 5.3(c) in order to demonstrate the relatively small degradation in the calibration accuracy at the extremes of the five stereo pairs. Each stereo pair is selected from three different sequences with Figure 5.3(b) being the ideal case. The smaller crosses indicate when the matches are those labelled as "good".



(a) frames 22 and 23 of 1st sequence



(b) frames 22 and 23 of 2nd sequence



(c) frames 38 and 39 of 3rd sequence

Figure 5.3: Examples of point correspondences from (a) 1st, (b) 2nd and (c) 3rd sequence.

Table 5.1 contains the results of five different examples where all three methods, labelled; *Or-warp* for feature orientation guided warping, *std-patch* for standard local area image correlation and *rot-patch* for the $\pm 4°$, rotated patch, were applied to three sequences (labelled '*Seq*') of stereo image pairs. Each set of five stereo pairs simulates when the calibration accuracy is both high and low. The frame numbers are given for reference only. The varying accuracy of calibration is achieved by

			Or-warp				*std-patch*				*rot-patch*			
Seq	*Frames*	F_{min}	M_f	M_g	M_b	P_M	M_f	M_g	M_b	P_M	M_f	M_g	M_b	P_M
	18:19	172	14	5	19	0.0	18	7	15	0.06	20	6	15	0.06
	19:20	173	50	3	3	0.28	55	4	3	0.32	58	3	2	0.34
1st	20:21	165	76	4	4	0.44	98	6	7	0.60	101	7	5	0.62
	22:22	137	54	3	1	0.41	77	2	2	0.56	77	1	2	0.57
	22:23	116	20	1	5	0.14	18	0	3	0.13	18	0	3	0.13
	20:21	166	13	3	16	0.0	11	2	13	0.0	9	3	12	0.0
	21:22	139	12	3	8	0.05	15	3	9	0.06	16	1	7	0.07
2nd	22:23	119	58	2	1	0.50	85	0	1	0.71	85	0	1	0.71
	23:24	106	60	2	2	0.57	81	0	1	0.75	81	0	1	0.75
	24:25	96	15	1	0	0.16	19	0	0	0.20	19	0	0	0.20
	38:39	149	9	4	13	0.0	12	4	16	0.0	12	4	16	0.0
	39:40	123	13	1	6	0.06	17	0	5	0.10	18	0	6	0.10
3rd	40:41	103	55	1	1	0.53	63	0	1	0.60	64	1	1	0.62
	41:42	92	58	0	2	0.61	67	0	0	0.73	67	0	0	0.73
	42:43	80	22	0	0	0.27	26	0	0	0.32	26	0	0	0.32

Table 5.1: Comparative results for point correspondence algorithms applied to uncorrected periscopic stereo images.

using a "weak" (not particularly accurate) calibration derived from a rotationally uncorrected image sequence using only the central, *fronto-parallel*[5], stereo image pair. As each subsequent stereo pair is selected, in either a clockwise or counter clockwise direction (in a rotationally uncorrected sense), from this central position the error in the position of the image data with respect to the epipolar constraint increases. Therefore the possibility of matching the corresponding image point is reduced in proportion to the displacement through the image sequence. This simulation of low calibration accuracy is explained further in Section 6.3.

Direct comparison of the results is not statistically significant. However, applying the performance measure to the data presented in Table 5.1 it is apparent that both the second and third techniques perform better than the first, since their scores are higher (compare the columns labelled 'P_M'), with the latter marginally better overall. The fact that correlating using the uncorrected image patches produced virtually identical results supports the concept that, in this particular situation, the 4° rotation

[5]implies the image plane is parallel to the scene with no rotation about the optical axis, see the glossary.

about the optical axis has a minimal effect.

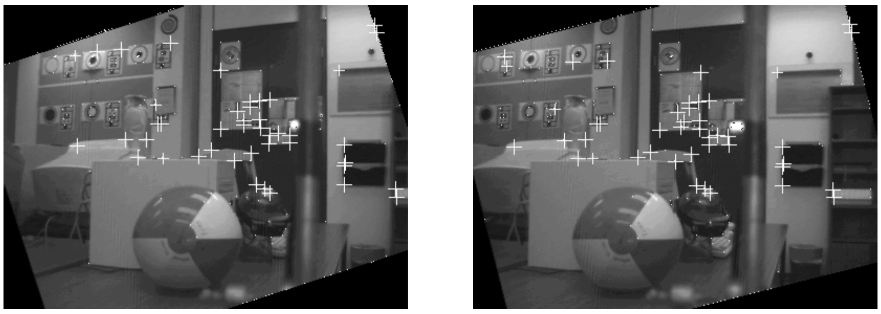
A similar test was conducted on a single, rotationally corrected, image sequence, derived from the 2$^{\text{nd}}$ sequence, using “more” stable calibration parameters. The calibration in this case is “more” stable due to the absence of the rotation about the optical axis. This is covered in Chapter 6. Table 5.2

		Or-warp				*std-patch*				*rot-patch*			
Frames	F_{min}	M_f	M_g	M_b	P_M	M_f	M_g	M_b	P_M	M_f	M_g	M_b	P_M
20:21	174	71	3	3	0.41	91	8	10	0.51	91	6	10	0.50
21:22	133	62	3	3	0.47	76	5	4	0.59	77	5	4	0.59
22:23	102	50	2	3	0.48	72	1	3	0.69	71	0	2	0.62
23:24	85	41	2	3	0.47	61	1	1	0.72	61	1	1	0.72
24:25	85	36	3	3	0.39	57	0	2	0.65	57	0	2	0.65

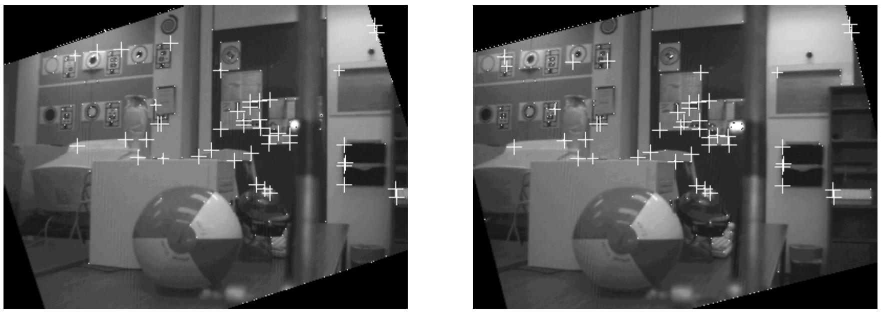
Table 5.2: Comparative results for point correspondence algorithms applied to rotationally corrected periscopic stereo images.

contains the results for this experiment and show that while the standard and rotated correlation patches identify more fixed matches, the number of bad matches is far greater than that for the orientation guided patch and/or the previous results. The reason for this is demonstrated in Figures 5.4(a) and 5.4(b) where the silhouette image frame in Figure 5.4(b) creates more corner features at the interface with the image data than the former. These features are naturally included by the correspondence algorithm and processed. This leads to a distinct reduction in the performance of the standard and rotated correlation patches but has a lesser effect on the orientation guided correlation patch. This is due to the fact that the feature orientation is not consistent along these contrived edges.

This anomaly also demonstrates the continuing problem of dealing with the silhouette frame throughout the remaining process chain. An obvious solution is to bound all processing by a *region of interest* (ROI) calculated from the size of the natural image frame and the frame number within the sequence. The concept of using a ROI is common place in many image processing environments,



(a) orientation guided correlation patch



(b) standard correlation patch

Figure 5.4: Point correspondence from region correlation using (a) orientation guided and (b) flat image patches on a rotationally corrected image sequence.

including `Tina`. However, the ROI must be the same for both images in the stereo pair. This restricts the maximum ROI to always be within silhouette frame $\pm 4^{\circ}$, which considerably reduces the viewing volume to about $\pm 1/4$ of the sequence from the cardinal point. Due to this limitation such a scheme has not been adopted.

In summary; the results of these experiments show that if the image sequence is rotationally corrected prior to correspondence matching then the warping of the correlation patch should be guided by feature orientation. If, however, the sequence is uncorrected the rotated patch should be used.

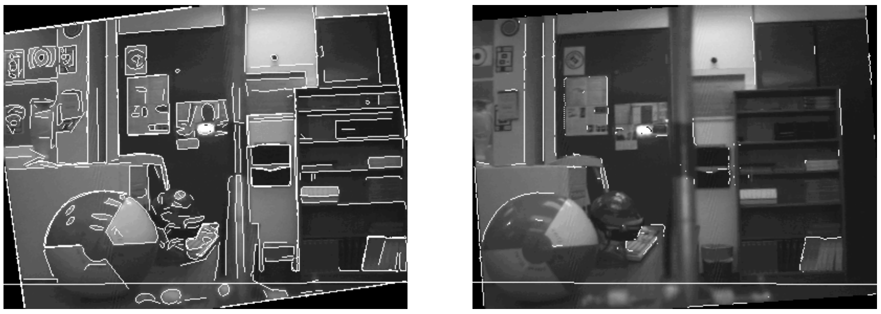
5.4 Correspondence for Reconstruction

Section 5.2 concentrated on solving the correspondence problem and producing a set of matched point features for subsequent re-calibration of a periscopic stereo imaging system. There also exists the requirement to provide a set of corresponding features for use in the estimation of the scene depth and subsequent reconstruction. The production of *depth maps*, or *disparity images*, is discussed in Chapter 7. Disparity images consist of either isolated point features or lines and conic sections where the depth estimate is encoded by the pixel value. `Tina` provides a number of stereo matching algorithms for the latter, edge string, features. These are briefly reviewed in this section, which has been included here in order to widen the overview of stereo correspondence algorithms and aid the discussion of future work in Section 5.5 and also later in Chapter 7.

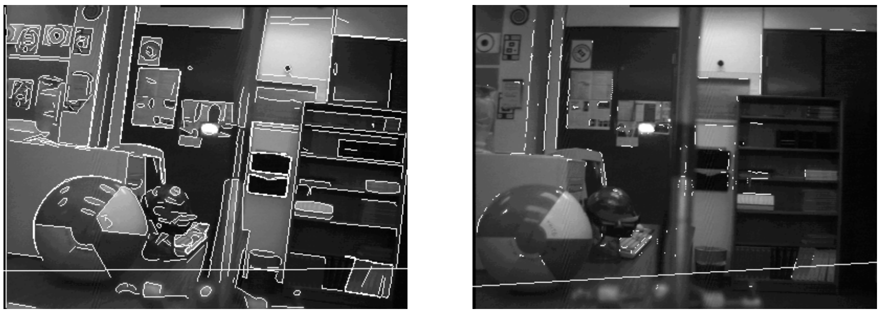
In Section 5.2 it was suggested that a small percentage of mismatches, or outliers, were permissible in the correspondence data because subsequent calibration should, in general, be sufficiently robust to accommodate their existence. However, if the correspondence data is required for reconstruction then no such relaxation should be tolerated. The correspondence algorithm must therefore include more stringent checks to reduce the possibility of outliers to a minimum.

A general feature-based correspondence algorithm is given in `Tina` which incorporates a number of options for match and support cost functions and list ordering to yield an optimal match list. Following image plane rectification using the row indexed method described in Section 5.2 and the calculation of a central disparity, a disparity window is created. This window constrains all candidate matches to fall within a valid range of disparity set by the central disparity estimate, the image dimensions and an upper and lower disparity threshold, all of which are set by external, user defined, parameters. A disparity histogram is then constructed from an initial match of edge strings accessed via the rectified row index of the left and right images. The matches are therefore consistent with the epipolar constraint. The disparity is calculated between the centroids of the matched strings. A selection of match options are available through `Tina's` tool interface. These are based on matching either 'Edgel' orientation, contrast, a combination of both or can include all features in the initial match set. The disparity histogram is then used to update the valid disparity range across the image, effectively implementing a disparity gradient constraint for later use. A list of matched edge strings is then constructed for both images based on the match function applied through the epipolar constraint, the disparity gradient and a uniqueness constraint implemented by labelling the matched strings. Match support structures are then added to the list of potential matches in order to keep track of the support for each match accumulated over the number of consistent elements in a whole string match process. Other support functions are available but the 'whole string match' support is recommended in `Tina`. The final stage of processing is the ordering of the match list to yield the preferred match. Three options are available. A best match at the string matching level, a *winner-take-all* scheme applied to all competing matches and a dynamic programming scheme which selects an optimal ordered match list by applying cost functions to the original constraints of the competing matches.

A detailed review of these methods has not been completed. However, the initial assessment of application to both rotationally corrected and uncorrected periscopic stereo image pairs has shown conflicting results. Figure 5.5 demonstrates some of the results of applying the same feature-based stereo algorithm to a rotationally corrected, as shown in Figure 5.5(a), and uncorrected, as shown in Figure 5.5(b), image pairs. In Figure 5.5(a) the match performance of the algorithm is good and there are few mismatches. However,, the inclusion of the silhouette frame does induce mismatch segments as shown on the right hand side of the bookcase and in the top left hand corner of the right hand image. The number of matched horizontal strings is naturally lower than non-horizontal strings due to the increased ambiguity of matching along the same raster line. In Figure 5.5(b) the match performance



(a) corrected periscopic stereo pair



(b) uncorrected periscopic stereo pair

Figure 5.5: Featured based stereo correspondence of (a) rotationally corrected and (b) uncorrected image pairs.

does not appear as good. Assuming the stereo calibration has an acceptable level of accuracy (as demonstrated by the inclusion of the epipolar lines in the Figures 5.5(a) and 5.5(b)) and considering that there is only a $4°$ difference between the feature orientation of corresponding edge strings, this feature based algorithm should be able handle the uncorrected images. It should be noted that only specific edge strings for matching since selecting every string would introduce more highlighted detail and make the visual comparison much more difficult. This apparent lack of performance in the case of the uncorrected images is a concern and should be the subject of further investigation.

It is known [QK96, MK98] that, in general, line features offer more flexibility to 3D reconstruction than point features and lead to more detailed models of the scene. These feature based stereo matching techniques are therefore important to large-scale scene reconstruction.

During the discussion of possible solutions to the correspondence problem in Section 5.2 the techniques, referred as stretch- and shear-correlation [LTM94], was mentioned. These techniques are able to produce correspondence matches of features, in spite of changes in feature profile, by employing a recursive search along the epipolar line with a range of warped correlation patches. The range of warping, which equates to variation in the width of the correlation patch, is reportedly [LTM94] consistent with enforcing a cyclopean disparity gradient limit along the epipolar line. In [LTM94] actual image plane rectification is applied, by interpolating the image data at positions specified by the camera's rectification matrix, prior to correlation. The algorithm also employs edge enhancement and filtering to pre-condition the image data prior to correlation of the warped patched. However, edge detection, is used to generate the positional information, to sub-pixel accuracy, required to construct disparity images based on non-horizontal edge strings. Non-horizontal edge strings are used in order to improve the stability of the algorithm which attempts to remove all ambiguous matches by using the disparity gradient limit.

The stretch-correlation method is, reportedly [LTM94] preferred and the technique was extended in [CLTS97] to incorporate a temporal, feedback loop. This involves the use of the previous disparity image to seed the matching process in the current frames. The use of this temporal information improves both reliability and efficiency by effectively reducing the search area along the epipolar line. Although edge strings provide the initial data set, only 3D point data is produced. No attempt was made to fit lines, or conic sections, to this 3D data.

The application of stretch correlation to rotationally corrected and uncorrected periscopic stereo images yields consistent results in both cases. The stretch- and shear-correlation algorithms were designed specifically for implementation in hardware and therefore are not optimized. The original paper [LTM94] concedes that the popular PMF algorithm [PMF85] is considerably more efficient. In the short term, use of the stretch-correlation algorithm for the production of correspondence data for periscopic stereo reconstruction appears limited due to its computational cost. However, its implementation in hardware could lead to real-time operation that would certainly be applicable to large-scale scene reconstruction by a mobile periscopic stereo system. The use of this temporal stereo algorithm is demonstrated in Chapter 7.

5.5 Concluding Remarks

In the introduction of this chapter it was stated that while solutions to correspondence problems may have common elements, they require "tailoring" to the particular set of circumstances relating to the imaging system, the type of corresponding feature and the subsequent process that receives the matched data set. This is evident in the discussion of the techniques developed for point correspondence and the techniques employed in `Tina` for both point and line correspondence algorithms.

This chapter has reviewed a number solutions to the stereo correspondence problem, applied to both point features and edge strings, with particular attention to their applicability to both rotationally corrected and uncorrected periscopic data. A simple technique, which simulates shear correlation using a $\pm 4^\circ$rotated patch, has been presented for the point correspondence in uncorrected periscopic stereo image pairs. Although this technique has not been conclusively tested, initial results show an overall level of performance better than the techniques reviewed. The rotated patch technique has therefore been adopted for the generation of corresponding point data for subsequent re-calibration of periscopic stereo using uncorrected image data. The technique recommended for point correspondence for rotationally corrected periscopic stereo image data is the feature orientation guided, warped correlation technique found in the `Tina` library. Further work is required to fully validate these techniques.

The review of techniques applicable to the generation of corresponding features required for subsequent reconstruction has been brief. Although there is some concern about the application of the featured based algorithm to rotationally uncorrected periscopic stereo image pairs, the techniques reviewed are capable of processing both corrected and uncorrected periscopic stereo image data. The stretch correlation algorithm is an established solution which is unaffected by the peculiarities of periscopic image data. The design appears optimal for matching corresponding features for subsequent reconstruction but less so for re-calibration. Both the featured based and stretch correlation techniques are referenced later in Chapter 7. Further work is required to validate the initial assessment of the applicability of these techniques to periscopic stereo.

Chapter 6

Calibration of Periscopic Stereo

Camera calibration is fundamental to the capabilities and performance of computer vision systems used for the 3D reconstruction of objects or scenes. The method of calibration directly affects the accuracy of the resulting model of the imaged scene and the type of reconstruction possible, as described in Chapter 2. This chapter applies two published techniques on camera calibration [Tsa87, TM91] to periscopic stereo and introduces a new emphasis on some of the key issues. Specifically, this chapter presents a new method for combining the two primary calibration techniques, grid calibration and epipolar calibration. The prescribed calibration algorithm includes a published technique [TM91], suitably modified, for updating the camera model over time. These methods apply to the calibration of both corrected and uncorrected periscopic data. This chapter begins with a review of the basic types of calibration in standard systems. Reviews of camera calibration techniques can be found at *CVOnline* or for more detailed treatments, refer to the texts by Faugeras [Fau93] and Hartley and Zisserman [HZ00].

Although a scaled Euclidean, or metric, reconstruction is possible with uncalibrated image sequences [PKVvG98], the cameras in a vision system should ideally be calibrated in order to achieve the best possible reconstruction results. In photogrammetric studies, camera calibration is referred to as solving the *orientation problems*. The two basic problems differentiate between recovering the internal (or *intrinsic*) and external (or *extrinsic*) parameters of the camera's *projection matrix*. The third problem referred to as *relative* orientation, recovers the relative transformation between the two camera views. The basic theory of the perspective camera model with intrinsic and extrinsic parameters, together with the geometry of *stereopsis* (two-view imaging) are given in Appendix A. The terms, *internal*, *external*, and *relative* camera parameters will be used throughout this chapter.

The particular method employed for the solution to the orientation, or camera calibration, problem depends on one of two possible configurations for the imaged scene.

- *Known scene*: where the position of a minimum of six points in the world and the corresponding 2D image points are known. These correspondences form a set of linear equations which can be solved and the camera parameters recovered by various forms of matrix decomposition. This is often referred to as *grid* calibration because of the use of a calibration object, or grid, placed in the scene which defines the world points.

- *Unknown scene*: where nothing is know about the scene, however, the corresponding image points from two or more views of the same world points are recovered to form a set of linear equations. These equations can be solved by either linear or non-linear methods and the camera parameters estimated *up to scale*. The term *up to scale* refers to the fact that there is no prior depth information in this configuration and therefore a normalized camera model is constructed without any absolute scale. A number of solutions are possible, each depending on the relative motion between the two camera positions. This motion constraint has two distinct possibilities:

– *Known camera motion*: which has three specific geometries:

  1. Pure translation.
  2. Pure rotation.
  3. Both rotation and translation.

– *Unknown camera motion*: the most general case, often termed *self-calibration* [FLM92].

Stereo calibration from an “unknown” scene can be described in general as *epipolar* calibration because of the explicit use of the epipolar constraint reviewed in Appendix A and used in the solution of the correspondence problem in Chapter 5. This term is preferred to *self-* or *auto-calibration* in this dissertation since it does not imply anything about the scene. The motive for this is explained in Section 6.4.

In Chapter 2 it was stated that full Euclidean reconstruction is only possible if some world reference is known. Therefore, solutions to the calibration from an unknown scene, regardless of the motion constraint, produce either restricted camera models and/or varying degrees of reconstruction ambiguity. That is either an affine camera that models parallel projection is assumed or only an affine reconstruction is possible. A number of solutions have been presented for the various camera configurations [PH95, MvGvDP93, Har94, HMDB95, MF92, ZDFL95] and types of scene and there is some diversity in the opinion of researchers concerning the preferred solutions given particular circumstances.

The relative camera motion between the views in periscopic stereo is “known”. This allows a useful simplification to the solution of epipolar calibration. Furthermore the use of a single camera yields fixed intrinsic parameters between stereo views. The aim of this chapter is therefore to investigate methods capable of producing full camera calibration for both rotationally corrected and uncorrected image pairs which will ultimately allow periscopic stereo to be used to estimate physical measurements of the scene.

Throughout this chapter reference will be made of the single camera and parallel camera models defined in `Tina`. The source code definitions of these are given in Appendix E for ease of reference.

6.1 Grid Calibration

The concept of relating 3D world points to 2D image points via *similar triangles* is discussed in Chapter 3 and in Appendix A. Knowing the coordinates of a set of n 3D world points and the corresponding images points, a set of linear equations can be formed such that,

$$\tilde{\boldsymbol{x}}_n = \mathrm{P}\tilde{\boldsymbol{X}}_n \tag{6.1}$$

where $\tilde{\boldsymbol{x}}_n$ and $\tilde{\boldsymbol{X}}_n$ are the image and world points in homogeneous coordinates and P is the camera projection matrix. The coordinates of the 3D world points are defined by the calibration target, or grid, and referenced to some chosen world origin. The most referenced solution for grid calibration found in the literature is that presented by Tsai [Tsa87].

6.1.1 Tsai's Method

First the coordinates of all the corresponding image points are converted into their equivalent camera coordinates as:

$$x_c = \frac{(x_i - u_0)d'_x}{s_x} \quad \text{and} \quad y_c = (y_i - v_0)d_y \tag{6.2}$$

where; u_0 and v_0 the coordinates of the principal point, s_x is a scaling factor which is initially set to one and derived explicitly later, d_y is the vertical distance between two sensor elements on the CCD array, d'_x is the horizontal distance between two adjacent pixels calculated from $d'_x = d_x N_{cx}/N_{fx}$, where d_x is the horizontal distance between adjacent sensor elements, N_{cx} is the number of sensor elements in a row and N_{fx} the number of pixels in the image row for the current resolution. N_{cx}

and d_x are often given in the camera's data sheet but can be derived by experiment where an image of a known square target is captured *fronto-parallel*[1] and the number of pixels is divide by the actual measurements of the square in millimeters. This is described in detail in Reg Willson's tutorial on Tsai's calibration[2]. This initial stage effectively defines the camera's internal parameters, excluding f, in Equations A.6 and A.7 in Appendix A and removes K from consideration in Equation A.8.

The next stage is to construct a matrix M from sets of linear equations of the form given by Equation 6.1. Considering only the Euclidean coordinates, the linear equations for transformation of 3D world points to 2D image points in the camera coordinate frame is given as:

$$\boldsymbol{x} = \begin{bmatrix} x_{c1} \\ x_{c2} \\ \vdots \\ x_{cn} \end{bmatrix} = \begin{bmatrix} y_{c1}X_{w1} & y_{c1}Y_{w1} & y_{c1}Z_{w1} & y_{c1} & -x_{c1}X_{w1} & -x_{c1}Y_{w1} & -x_{c1}Z_{w1} \\ y_{c2}X_{w2} & y_{c2}Y_{w2} & y_{c2}Z_{w2} & y_{c2} & -x_{c2}X_{w2} & -x_{c2}Y_{w2} & -x_{c2}Z_{w2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{cn}X_{wn} & y_{cn}Y_{wn} & y_{cn}Z_{wn} & y_{cn} & -x_{cn}X_{wn} & -x_{cn}Y_{wn} & -x_{cn}Z_{wn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_7 \end{bmatrix} = \mathrm{M\,A} \tag{6.3}$$

where $\boldsymbol{A}$ is the 7-vector solution, from which the external camera parameters are derived, and M is an n by 7 matrix defining the problem space. This only applies if the calibration points are derived from a non-coplanar calibration grid. If the calibrations points are all coplanar then the terms containing Z_w are removed from the problem space, as recommended in [JKS95, KSK98, Wil94], thereby reducing $\boldsymbol{A}$ to a 5-vector. The missing parameters must then be determined indirectly, as described later.

The use of more than seven points (continuing for the non-coplanar case) leads to an over-determined set of equations which can be solved for the vector A using a number of methods:

- The pseudo-inverse technique (or *Moore-Penrose* extension of the generalized inverse A^{+})

[1] see the Glossary for terms which are in *italics*.

[2] Reg Willson's tutorial and source code is available from:

`http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html` where,

$$\boldsymbol{A} = (\mathrm{M}^T\mathrm{M})^{-1}\mathrm{M}^T\boldsymbol{x} \tag{6.4}$$

is recommended[3] in [KSK98].

- Single Value Decomposition (SVD)[4] where M can be decomposed into the product of three matrices such that,

  $$\mathrm{M} = \mathrm{U\,D\,V}^T \tag{6.5}$$

  where U has orthonormal columns, D is a non-negative diagonal and V^T has orthonormal rows, is recommended in [JKS95, SHB99].

- The method implementation in `Tina` uses a *Cholesky* least-square solution, which is similar to *LU* decomposition for a specialized case of a symmetric, positive-definite matrix and ensures a positive, non-singular solution [PFTV93, sect:2.9].

From the analysis in [Tsa87] and [KSK98], the parameters of the solution vector equate to:

$$a_1 = \frac{r_1 s_x}{t_y},\quad a_2 = \frac{r_2 s_x}{t_y},\quad a_3 = \frac{r_3 s_x}{t_y},\quad a_4 = \frac{t_x s_x}{t_y},\quad a_5 = \frac{r_4}{t_y},\quad a_6 = \frac{r_5}{t_y},\quad a_7 = \frac{r_6}{t_y} \tag{6.6}$$

where t_x and t_y are two of three components of the translation vector $\boldsymbol{t}$, r_j are components of the rotation matrix Rwhich form $[\,\mathrm{R}\,|\,\boldsymbol{t}\,]$, the transformation, or projection matrix, from the world to the camera coordinate frame. Using the orthonormal property of R $(r_x^2 + r_y^2 + r_z^2 = 1)$ and parameters a_4, a_5 and a_6 a value for t_y can be calculated. Initially the sign of t_y is assumed to be positive and r_1, r_2, r_4, r_5 and t_x are calculated from Equations 6.6. Then the signs of the components of the most eccentric image point in the calibration data set are compared to the signs of:

$$x' = r_1 X_w + r_2 Y_w + t_x \qquad \text{and} \qquad y' = r_4 X_w + r_5 Y_w + t_y \tag{6.7}$$

[3] This is also used in the implementation by Reg Willson.

[4] The theory and code for SVD can be found in *Numerical Recipes* [PFTV93, sect:2.6]

If the signs are the same then all the calculated parameters remain positive, if not then all the signs are set negative.

The scaling factor s_x is determined using the orthonormal property of R and the fact that image scanning orientation is assumed to be from left to right (s_x always positive) such that:

$$t_y\sqrt{a_1^2+a_2^2+a_3^2} = t_y\sqrt{(t_y^{-1}s_x r_1)^2+(t_y^{-1}s_x r_2)^2+(t_y^{-1}s_x r_3)^2} = s_x\sqrt{r_1^2+r_2^2+r_3^2} = s_x \quad (6.8)$$

If only five parameters are available, from the use of coplanar calibration data, then t_y is determined from:

$$t_y = \frac{1}{\sqrt{a_1^2+a_2^2+a_4^2+a_5^2}} \quad (6.9)$$

The remaining parameters are calculated as described above except for the scaling factor s_x which can not be calculated in such circumstances. It is therefore ignored until a later stage of processing. If the scaling factor is available then the components, $r_1,\ r_2,\ \ \ldots\ \ ,r_6$, of the rotation matrix are finally calculated using Equations 6.6. If not then the orthonormal property of R is used again to determined r_3 and r_6. In either case, the last row, $\boldsymbol{r}_c$, of the rotation matrix is determined from the other two rows by, $\boldsymbol{r}_c = \boldsymbol{r}_a \times \boldsymbol{r}_b$

An initial estimate of the focal length and the t_z component of the translation vector is determined from the set of linear equations:

$$\begin{bmatrix} g_1 & y_{c1} \\ g_2 & y_{c2} \\ \vdots & \vdots \\ g_n & y_{cn} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} h1 & y_{c1} \\ h2 & y_{c2} \\ \vdots & \vdots \\ hn & y_{cn} \end{bmatrix} \quad (6.10)$$

formulated from n calibration points where,

$$g_j = r_4 X_{wj} + r_5 Y_{wj} + t_z \quad \text{and} \quad h_j = r_7 X_{wj} + r_8 Y_{wi} \quad (6.11)$$

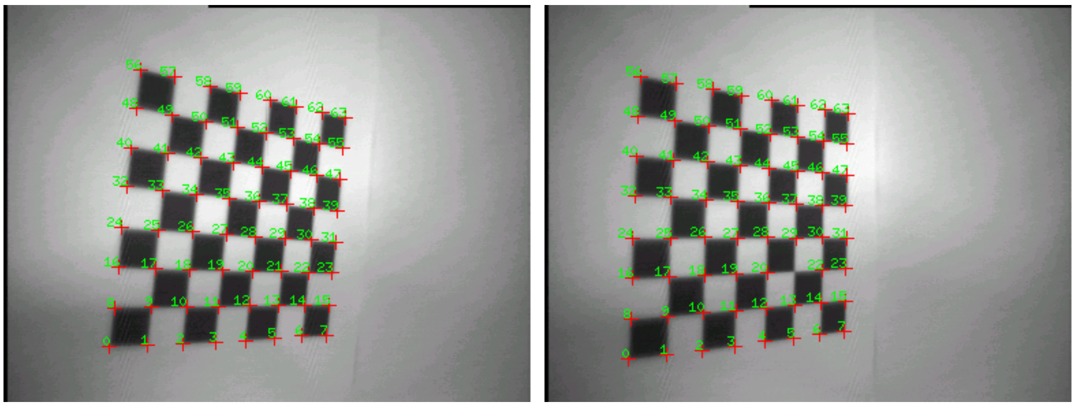
This n by 2 matrix is solved for f and t_z as above, using the same preferred method. The estimates for f and t_z are improved by constrained optimization. This process can incorporate estimates of radial lens distortion as described in [JKS95, KSK98, SHB99, HZ00, Tsa87]. The implementation given by Reg Willson includes optimization with 1st order radial lens distortion. The implementation given in `Tina` consists of a separate process which does not, as yet, include radial lens distortion. The separate process does, however, allow for a "global" optimization of any or all of the internal camera parameters and all of the external camera parameters using the simplex optimization method [PFTV93]. The error function used for the optimization computes the sum of the square error of the corresponding calibration points projected onto the image plane. This is given by Equation 6.12 and discussed in more detail later. No weighting factors are used and all the parameters selected are optimized without bias. The scaling factor in the original algorithm, can be "partially" compensated in the optimization process by the selection of one of the aspect ratio parameters together with principal point u_0, v_0 (labelled a_x, a_y, c_x and c_y in `Tina`)[5], and the focal length f. However, considering that all parameters are optimized equally in simplex optimization and s_x applies to some parameters and not others, see Equation 6.6. This is a source of error which impacts on the performance of this implementation and identifies a limitation with using coplanar calibration data.

6.1.2 The Design of Calibration Grids

This section discusses the design of calibration grids. As mentioned in Section 6.1.1, the grid calibration can consist of either coplanar or non-coplanar calibration points. Examples of non-coplanar calibrations grids are given in the referenced texts [Fau93, HZ00, chap.6,pp.170] Coplanar grids are generally of the form shown in Figure 6.1. Non coplanar calibration grids are usually formed by two

[5]The aspect ratio parameters effectively scale the horizontal and vertical axis of image plane according to dimensions of the pixel grid and the current image resolution. In practice only the horizontal a_x is ever used.

coplanar grids at 90°to each other. Identification of the corresponding image points requires feature extraction to sub-pixel accuracy together with a subsequent *labelling* procedure which identifies and matches specific points in an ordered fashion. The result of such labelling is also demonstrated in Figures 6.1(a)–6.1(b). It is essential that the techniques employed to identify the corresponding image



(a) frame 11 (b) frame 12

Figure 6.1: Successive frames of a coplanar calibration grid inside a small box.

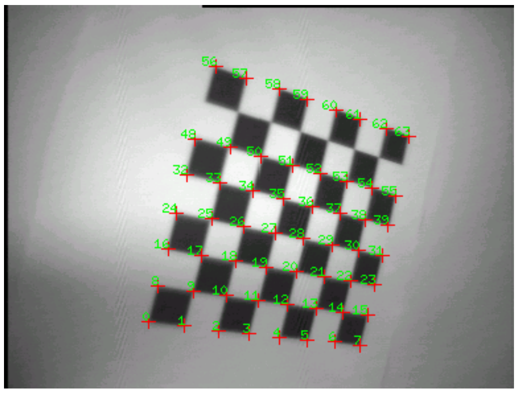
points are invariant under the perspective distortion created by the camera. Due to the tumbling nature of the uncorrected image data between successive frames from periscopic stereo, the techniques employed should also be rotationally invariant. This does not present a problem for the feature extraction part, which consists of edge detection followed by line fitting; the calibration points are recovered to sub-pixel accuracy from the intersection of the lines. However, robust labelling and correspondence matching is more difficult.

Labelling on coplanar grids is simpler than that on non-coplanar grids. The reasons for this are given in the next paragraph. Unfortunately coplanar grids are not the preferred choice in the referenced texts [Fau93, HZ00, KSK98]. The primary reason for this is that for many of the optimization methods used in the various calibration techniques coplanar data points yield degenerate cases. The is discussed in [HZ00, sect:10.9] for the estimation of the fundamental matrix using the *8-point al-*

gorithm and herein Section 6.1.1, page 113, where the use of the Z_w coordinates are precluded for coplanar calibration points.

Non-coplanar calibration grids tend to be larger than their coplanar counterparts. This is due, in part, to the desire for a greater number of calibration points for an over-determined set of linear equations. However, more important is the increased ambiguity in labelling the corresponding image points for both the left and right planes. One method of reducing the ambiguity in labelling the corresponding image points is to employ the concept of the *cross-ratio* which remains invariant under projective transformations [Fau93, HZ00], as described in Chapter 2 on page 18. Using the cross-ratio, sets of four collinear image points can be matched to their corresponding set of four collinear calibration points. By using different sets of four points an ordered and labelled list of points can be produced. This is the method employed in `Tina` and that used to derive the labelling shown in Figure 6.1.

It was suggested in Chapters 1 and 3 that the primary application of large-scale reconstruction from periscopic stereo would be for remote operation in hostile environments. The idea using a calibration grid placed in the scene is therefore counter-productive. However, by using a calibration grid placed in the corner, inside a small small box, this apparent limitation can be over come. The box is placed over the periscopic stereo head. This “calibration in a box” idea allows the system to be calibrated prior to use. However, its use introduces a limitation to grid size which must be completely visible in a single image frame. The author has made attempts to incorporate non-coplanar calibration grid (across the corner of the box) and maintain the use of the cross-ratio. This requires a total of at least six collinear calibration points on the grid in order to solve the labelling ambiguity. Reducing the number of calibration points on each half of non-coplanar grid leads to an increased likelihood of incorrect labelling especially given the tumbling image data. Figure 6.2 demonstrates that even with grid of 8×8 calibration points, errors still occur. Notice the top rows in Figure 6.2(a) and the left


(a) frame 9

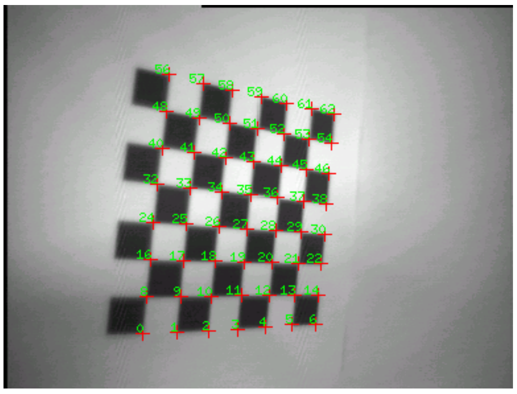

(b) frame 11

Figure 6.2: Incorrectly labelled calibration points

hand columns in Figure 6.2(b). Figures 6.1(a) and 6.2(b) are of the same frame using different data files which define the location of the corners of each square on the grid in the world coordinates. The only difference in these files is the location of the world origin from where the calibration points are defined. This should not, in theory, affect the labelling process which computes the extended grid lines from top to bottom and then labels the intersections from bottom left to top right as shown in Figures 6.1 and 6.2.

At the time of writing, an optimal solution has not been found. The coplanar calibration grid, with 8×8 calibrations points, as shown in Figures 6.1 and 6.2 is used, with specific data files known to yield no labelling errors over these frames, for the initial camera calibration described in this chapter. The source code for the recovery of the calibration data used in the experiments presented herein is the same as that available in `Tina`. The design of a non-coplanar calibration grid for use inside the box requires further work to yield an optimal compromise between size, complexity and robust recovery of the calibration points.

6.1.3 Problems with Tsai's Method

Although Tsai's method is widely referenced, there are a number of limitations and sources of error which are not always conveyed to the reader. Two constraints for the use of Tsai's method are recommended in [JKS95]. These are:

1. The world origin in the absolute coordinates is not in the field of view.
2. The world origin does not project to a point in the image that is close to the *y* axis of the camera coordinate frame.

Condition 1, reportedly [JKS95] decouples the effects of radial lens distortion from the focal length and the distance to the calibration grid. Condition 2 ensures that the t_y component of the translation vector is not close to zero and therefore does not present a problem in Equation 6.6. In practice neither of these constraints present a problem since a world origin can be offset from the calibration grid by some translation, with the data points scaled accordingly. For the experiments in this section the world origin was chosen as the base of the corner of the box plus an offset translation vector of $[\,250, 100, 0\,]$. The calibration data points were specified assuming a *fronto-parallel* condition such that the Y_w coordinates of the horizontal collinear calibration points are all the same.

In section 6.1.1, the review of Tsai's method reveals that the initial phase of processing requires the corresponding image points to be converted to the camera coordinate frame using an initial "guess" of the principle point and the aspect ratio of the camera's pixel grid. The initial values, recommended by Tsai [Tsa87], assume the principal point to be at half the image width and height and the aspect ratio to be 1:1. The initial focal length is determined by an "educated" guess, based on the lens parameters. Although improved estimates of the principal point, aspect ratio and focal length can be determined by subsequent optimization, as is the case in `Tina's` implementation, their requirement presents a distinct limitation if "good" initial estimates are not available. This fact has been recognized in [JKS95, KSK98]. However, no evidence is given in these texts to the extent of the effect of "bad" initial guess of these parameters. What constitutes "good" and "bad" initial estimates of the principal point, or indeed, the other internal camera parameters, appears from the literature, to be resolved by "trial and error". This impacts on the use of the implementation given in `Tina`. Since the final optimization phase considers all parameters to have equal error, the result can be an optimal solution with large accuracies in any particular parameter. In order to assess these effects a number of attempts to calibrate the camera using the same data, associated with Figure 6.1(b), were conducted while varying the initial estimates of the principal point, u_0 and v_0. The camera is assumed to be coincident with the world origin such that a zero translation vector is specified and the rotation matrix is defined by an identity matrix. The initial estimate of focal length was set to 10 and the aspect ratios $a_x = 1.1,\ a_y = 1.0$. The results are given in Table 6.1, where f_0 is the initial value of focal length returned by the basic Tsai algorithm and f_{min} is the estimate of the focal length after optimization. $\sum \varepsilon^2\, IP_{min}$ is the sum of the squared error function for minimizing the difference between the corresponding image and world ($\boldsymbol{X}$) calibration points:

$$\sum \varepsilon^2\, IP_{min} = \sum \left((X_u - x_u)^2 + (X_v - x_v)^2 \right) \tag{6.12}$$

projected on to the image plane (u,v). The use of the sum of squared error for optimization is a valid measure of the accuracy of result only because the number of calibrations points remains constant throughout this experiment. The central pixel (192, 144 - labelled '#') was chosen as the origin of an iterative search and the variation of u_0 and v_0 was chosen to form a "star" pattern in the centre of the image. Table 6.0(a) presents the results for a single-stage optimization process and Table 6.0(b) the results for a two-stage process where the central columns of show the optimization with only the focal length included with the external camera parameters. The five columns on the right-hand side of both Tables 6.0(a) and 6.0(b) show the results of optimization with u_0, v_0, a_x and f_{min} included

(a) Single-stage optimization

	Guess P & Tsai f			$IP_{min}(\, f_0,\, u_0,\, v_0,\, a_x\,)$					
start	u_0	v_0	f_0	$\sum \varepsilon^2 \, IP_{min}$	u_0	v_0	a_x	f_{min}	*end*
	184	139	1.37	11.393	213.6	104.1	1.51	6.31	?
	192	139	1.58	12.393	161.6	119.9	1.25	5.54	
	200	139	1.78	7.096	223.9	110.6	4.03	7.10	
	184	144	1.30	‡ 9.6e+08	187.9	138.3	1.09	1.33	
#	192	144	1.54	8.884	70.7	129.3	1.21	4.56	?
	200	144	1.74	11.972	207.9	109.7	1.40	6.15	
	184	149	1.23	14.546	207.9	120.9	1.18	5.69	
	192	149	1.49	23.934	216.8	152.9	0.85	4.04	
	200	149	1.71	20.263	252.0	134.3	0.97	5.32	

(b) Two-stage optimization

	Guess P & Tsai f			$IP_{min}(\, f_0\,)$		$IP_{min}(\, f_{min},\, u_0,\, v_0,\, a_x\,)$					
start	u_0	v_0	f_0	$\sum \varepsilon^2 \, IP_{min}$	f_{min}	$\sum \varepsilon^2 \, IP_{min}$	u_0	v_0	a_x	f_{min2}	*end*
	184	139	1.37	17.368	5.12	16.090	188.5	129.1	1.08	5.18	$*$
	192	139	1.58	17.946	5.28	16.427	196.6	128.5	1.08	5.26	
	200	139	1.78	18.230	5.35	17.791	200.7	133.6	1.02	5.03	
	184	144	1.30	19.520	5.20	13.564	176.7	121.7	1.20	5.51	
#	192	144	1.54	20.271	5.27	18.088	214.2	132.6	1.02	5.14	
	200	144	1.74	20.483	5.35	17.303	222.4	128.5	1.05	5.38	
	184	149	1.23	22.511	5.21	16.952	193.6	131.0	1.06	5.11	
	192	149	1.49	23.251	5.28	15.990	205.0	126.4	1.10	5.40	
	200	149	1.71	23.966	5.36	16.637	199.4	129.2	1.07	5.24	

Table 6.1: Initial results for Tsai calibration with varying estimates of the principal point.

with the external camera parameters. The former produces less stable results, as indicated by a wide variation in optimized estimates of the internal camera parameters, large ($\geq$*20.0*) errors and a failure to minimize after a second attempt (labelled '‡') and there is no clear indication (labelled '?') for the best direction of the true location of the principal point. The initial optimization should therefore always be conducted for the focal length only. The results from the second optimization, far right-hand columns of Table 6.0(b), show a range of estimates for the camera parameters, some more "stable" than others since the final estimates of the parameters are not widely disparate, or illogical, for instance

$2.0 \gg a_x \ll 1.0$.

A similar experiment was conducted for an aspect ratios of $a_x = 1.0,\ a_y = 1.0$. The results are not given here in order to save time and space. However, these were, in general, less stable than those given in Table 6.1. The fact that more stable results are achieved with a greater horizontal than vertical scaling is logical since both the image and the camera's CCD array have a greater width than height. However, the inclusion of the horizontal scaling at an early stage in the iterative optimization process leads to less stable results, as shown in the central columns (results labelled '†') of Table 6.1(a). The best result achieved (combination of lowest error and most stable estimates of quoted parameters) is highlighted (labelled '$*$') in Table 6.0(b) and was chosen for the centroid of a second set of calibrations with a smaller variation of the coordinates from the estimated principal point. The experiment was repeated twice, each time selecting the most favorable result in an attempt to converge to an optimal result via a sort of crude gradient decent approach. Table 6.2 contains the results from third set of calibrations.

The results in Table 6.2 are arranged in a similar manner to Table 6.1 except for the inclusion of an intermediate optimization stage for f_0 and a_x as shown in Table 6.1(a). Optimization of all the parameters is conducted after (Table 6.1(a)), or instead of (Table 6.1(b)), the intermediate stage. A star search pattern is used with the inclusion of two extra calibration tests at (167,139) and (168,139). Although the "guess" coordinates of the principal point now lay within a 3×8 pixel region of the previous "best" result, the final results of parameter optimization still yield some erratic values. For an initial estimate of (168,139) for the principle point no minimal solution is achieved (labelled '‡'). While for the adjacent pixel (167,139) a minimal solution with "apparently" satisfactory estimates of the internal camera parameters is achieved. It should be noted that the implementation in `Tina` contains an escape condition if a solution is not forthcoming after a given number optimization attempts. Excluding these two results, all but two other (labelled '†') of the final results in Table 6.1(a) appear

(a) Three-stage optimization

	Guess P & Tsai f			$IP_{min}(\, f_0\,)$		$IP_{min}(\, f_{min},\, a_x,\,)$			$IP_{min}(\, f_{min2},\, u_0,\, v_0,\, a_x\,)$					
start	u_0	v_0	f_0	$\sum \varepsilon^2\, IP_m$	f_{min}	$\sum \varepsilon^2\, IP_m$	a_x	f_{min2}	$\sum \varepsilon^2\, IP_m$	u_0	v_0	a_x	f_{min3}	*end*
	167	139	0.92	15.900	5.03	16.844	1.00	4.50	16.863	167.0	139.0	1.00	4.50	
	168	139	0.94	‡ 5.6e+09	‡ 1.04	‡ 6.4e+06	1.15	1.34	14.906	161.6	131.9	1.08	4.89	?
	172	139	1.03	16.282	5.08	16.331	1.10	5.08	15.066	172.2	129.7	1.10	5.08	
	176	139	1.14	16.615	5.12	17.125	1.02	4.70	17.029	177.4	136.2	1.02	4.73	
	180	139	1.26	17.022	5.16	17.497	1.01	4.70	17.209	178.1	136.8	1.01	4.71	
	172	140	1.01	16.614	5.08	16.653	1.10	5.08	14.870	176.6	127.5	1.12	5.21	?
#	176	140	1.12	16.985	5.12	18.189	† 0.98	4.43	18.217	175.9	140.1	† 0.98	4.43	
	180	140	1.24	38.190	5.32	12.335	† 2.46	6.86	12.280	180.0	140.6	† 2.45	6.85	
	172	141	0.98	17.013	5.08	17.031	1.10	5.08	16.179	184.6	130.5	1.07	5.09	
	176	141	1.10	17.322	5.12	17.244	1.10	5.13	15.053	180.5	127.2	1.12	5.25	
	180	141	1.22	17.761	5.16	17.575	1.10	5.17	14.481	171.4	127.4	1.12	5.19	?

(b) Two-stage optimization

	Guess P & Tsai f			$IP_{min}(\, f_0\,)$		$IP_{min}(\, f_{min},\, u_0,\, v_0,\, a_x\,)$					
start	u_0	v_0	f_0	$\sum \varepsilon^2\, IP_{min}$	f_{min}	$\sum \varepsilon^2\, IP_{min}$	u_0	v_0	a_x	f_{min2}	*end*
	167	139	0.92	15.900	5.03	15.978	178.7	131.4	1.70	5.01	$*$
	168	139	0.94	‡ 5.6e+09	‡ 1.04	11.766	90.6	150.5	† 0.98	2.86	
	172	139	1.03	16.282	5.082	15.324	176.6	129.4	1.09	5.11	
	176	139	1.14	16.615	5.12	15.126	168.6	131.0	1.09	4.10	?
	180	139	1.26	17.022	5.16	16.428	184.7	131.4	1.06	5.04	
	172	140	1.01	16.614	5.08	16.930	190.5	131.4	1.05	5.06	
#	176	140	1.12	16.985	5.12	15.649	186.8	128.0	1.10	5.23	
	180	140	1.24	38.190	5.32	24.242	194.2	152.3	1.07	5.21	
	172	141	0.98	17.013	5.08	16.600	189.4	130.9	1.06	5.10	
	176	141	1.10	17.322	5.12	16.094	180.6	131.3	1.70	5.09	
	180	141	1.22	17.761	5.16	14.625	168.0	128.7	1.11	5.10	

Table 6.2: Search results for Tsai calibration with varying estimates of the principal point.

plausible. As stated above. these results appear less stable and there is no clearly “best” result from the three-stage optimization. Although a “best” result (labelled ‘$*$’) is highlighted in Table 6.1(b), the choice is based more on the author’s own definition of “stable” rather than on a definitive solution.

Collectively the results given in Tables 6.1 and 6.2 clearly demonstrate the fundamental problem with Tsai’s method. Although a solution can be achieved after a number of iterations, “stable” and logical values for the camera parameters are not guaranteed and the number of iterations is dependent on the initial guess. It is apparent that if the initial guess is inaccurate by more than a few pixels, the iterative optimization will yield estimates of the internal camera parameters, in particular the coordinates of the principle point, that oscillate about the optimal result. This is partially compensated in the implementation in `Tina` with the inclusion of maximum error bounds in the error function of the simplex optimization process which effectively enforces a slow, steady convergence to the result. However, the question of how many iterations are required for an optimal solution, or the validity of values in that solution, is still ambiguous. This is not the fault of the simplex optimization process. Even if the camera parameters were weighted, or bounded, in some way, a poor initial guess would still be liable to yield the sort of instability demonstrated here.

Given the less than desirable nature of the results from these experiments, it is difficult to see how the optimal conditions could be modelled in order to generate a cost function which could be used to automate the search process conducted in these experiments. The inclusion of Tsai's algorithm in an automated calibration process is therefore highly unlikely. It should be noted that the results also show that the coordinates of the principle point for the camera used in these experiments are some considerable distance, in pixels, from the assumed centre of the image. In fact, the best results were recorded in a 3×3 pixel region surrounding an initial estimate of the principle point at (162, 138). This is not uncommon in low-cost CCD cameras [Wil94] but far greater than expected here. In spite of the shortcomings of Tsai's method an initial, user assisted, calibration is possible for both rotationally corrected and uncorrected periscopic stereo image pairs.

6.1.4 Alternative Method of Grid Calibration

Recently, an alternative method for grid calibration has arisen. This method appears in [SHB99] and is covered in detail in [HZ00]. The following is a brief outline of the method.

From Equation 6.1, the 3×4 projection matrix is parameterized such that,

$$\begin{bmatrix} \lambda x_i \\ \lambda y_i \\ \lambda \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11}X_w + p_{12}Y_w + p_{13}Z_w + p_{14} \\ p_{21}X_w + p_{22}Y_w + p_{23}Z_w + p_{24} \\ p_{31}X_w + p_{32}Y_w + p_{33}Z_w + p_{34} \end{bmatrix} \tag{6.13}$$

where λ is the arbitrary homogeneous scale and is removed to yield two linear equations:

$$\begin{aligned} x_i\left(p_{31}X_w + p_{32}Y_w + p_{33}Z_w + p_{34}\right) &= p_{11}X_w + p_{12}Y_w + p_{13}Z_w + p_{14} \\ y_i\left(p_{31}X_w + p_{32}Y_w + p_{33}Z_w + p_{34}\right) &= p_{21}X_w + p_{22}Y_w + p_{23}Z_w + p_{24} \end{aligned} \tag{6.14}$$

For n calibration points a $2n \times 12$ matrix is constructed such that:

$$\begin{bmatrix} X_{w1} & Y_{w1} & Z_{w1} & 1 & 0 & 0 & 0 & 0 & -x_{i1}X_{w1} & -x_{i1}Y_{w1} & -x_{i1}Z_{w1} & -x_{i1} \\ 0 & 0 & 0 & 0 & X_{w1} & Y_{w1} & Z_{w1} & 1 & -y_{i1}X_{w1} & -y_{i1}Y_{w1} & -y_{i1}Z_{w1} & -y_{i1} \\ & & \vdots & & & & & & & \vdots & & \\ X_{wn} & Y_{wn} & Z_{wn} & 1 & 0 & 0 & 0 & 0 & -x_{in}X_{wn} & -x_{in}Y_{wn} & -x_{in}Z_{wn} & -x_{in} \\ 0 & 0 & 0 & 0 & X_{wn} & Y_{wn} & Z_{wn} & 1 & -y_{in}X_{wn} & -y_{in}Y_{wn} & -y_{in}Z_{wn} & -y_{in} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ \vdots \\ p_{34} \end{bmatrix} = 0 \tag{6.15}$$

or A$\boldsymbol{P} = 0$.

A minimum of six corresponding calibration points are required to solve Equation 6.15. More calibrations points lead to an over-determined set of linear equations which can be solved using a robust least-squares method, subject to $\|\boldsymbol{P}\| = 1$. Singular Value Decomposition (SVD) [PFTV93] is recommended in [HZ00] for the solution Equation 6.15, where $\mathrm{A} = \mathrm{UDV}^T$, with the positive diagonal of D arranged in descending order, then $\boldsymbol{P}$ is given by the last column of V. In [HZ00] further recommendation is given to use data normalization, proposed by [Har95], to precondition both the world and image points thereby leading to more stable solutions when using SVD. The normalization does not affect the accuracy of the result and is reversed after a satisfactory result is achieved. The result from this linear solution can then be used as an initial estimate for an iterative optimization of $\boldsymbol{P}$ using Levenberg-Marquardt [PFTV93], or an alternative method, while minimizing the geometric error given by Equation 6.12. Correcting for lens distortion can be included in the optimization process, as mentioned in Section 6.1.1.

The external and internal camera parameters are recovered from $\boldsymbol{P}$ by exploiting the fact that:

$$\boldsymbol{P} = [\,\mathrm{KR}\,|\,-\,\mathrm{KR}\boldsymbol{t}\,] = [\,\boldsymbol{M}\,|\,\boldsymbol{b}\,] \tag{6.16}$$

where $\mathrm{M} = \mathrm{KR}$ is a 3×3 sub-matrix and the translation vector is given by $\boldsymbol{t} = -\mathrm{M}^{-1}\boldsymbol{b}$. Since the rotation matrix R is orthogonal and the camera matrix K is upper triangular, as shown in Equation A.8 of Appendix A, both can be recovered using *QR* matrix factorization [PFTV93].

This grid calibration method, referred to as a *gold standard* algorithm in [HZ00], is worth further investigation, especially into its use with a periscopic stereo system and is referenced later in Section 6.5.

6.2 Epipolar Calibration

The concept of camera calibration without the use of a calibration target or some other known world data, referred to as self-, or auto-calibration [FLM92], is based on the deterministic nature of the relative geometry between the two views of stereo imaging system. An introduction to two-view, or epipolar, geometry is given in Appendix A. This section briefly reviews the popular methods recommended in the referenced texts.

The epipolar geometry between two views is defined by the *fundamental* matrix such that:

$$\tilde{\boldsymbol{x}}^T \mathrm{F}\, \tilde{\boldsymbol{x}}' = \begin{bmatrix} x_i, & y_i, & 1 \end{bmatrix} \mathrm{F} \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = 0 \tag{6.17}$$

where F is a 3×3 matrix which can only be determined up to scale. Multiplying out and rearranging, Equation 6.17 can be written in the form of a vector inner product. If n corresponding images points are available, the vector inner product can be expanded to form a set of linear equations in matrix form:

$$\mathrm{A}\boldsymbol{f} \begin{bmatrix} x_{i1}x'_{i1} & x_{i1}y'_{i1} & x_{i1} & y_{i1}x'_{i1} & y_{i1}y'_{i1} & y_{i1} & x'_{i1} & y'_{i1} & 1 \\ & \vdots & & & & & & \vdots & \\ x_{in}x'_{in} & x_{in}y'_{in} & x_{in} & y_{in}x'_{in} & y_{in}y'_{in} & y_{in} & x'_{in} & y'_{in} & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{33} \end{bmatrix} = 0 \tag{6.18}$$

Since $\boldsymbol{f}$ can be determined, at best, up to scale, a unique linear solution is possible for eight corresponding, error free, image points. If error exists in the positional estimates of the data then a least-squares solutions is required. This is referred to as the 8-point algorithm. The basic concept of computing the fundamental matrix is accredited to Longuet-Higgins [LH81]. However, the original implementation was applied to calibrated cameras, thereby yielding the essential matrix, as described in Appendix A.

The recommended [HZ00, SHB99] solution for Equation 6.18 is to use SVD, minimizing $\|\mathrm{A}\boldsymbol{f}\|$ subject to $\|\boldsymbol{f}\| = 1$. This is similar to the use of SVD mentioned in Section 6.1.4, only with a reduced problem space. An important property of the fundamental matrix is that it is singular and of rank 2 [HZ00]. The initial solution derived from SVD does not enforce this constraint. However, matrix F can be replaced by a matrix F$'$ that minimizes the Frobenius norm $\|\mathrm{F} - \mathrm{F}'\|$ subject to $\det \mathrm{F}' = 0$. This is achieved by applying SVD to F. The recommended algorithm, incorporating data normalization, as described in [HZ00]. A minimum case solution, requiring only seven corresponding image points, is possible using the constraint $\det \mathrm{F} = 0$. This is also described in [HZ00].

The 8-point algorithm yields a direct solution but is sensitive to positional errors in the data, which was the motivation behind the inclusion of data normalization in [Har95]. The algorithm will also fail if there are *outliers* present in the data. A number of robust optimization methods are possible together with a number of possible error functions. [HZ00] recommends the Levenberg-Marquardt method of optimization, minimizing with either an algebraic error function or a ‘Maximum Likelihood’ (ML) estimator that minimizes the geometric distance of the re-projected points on the image plane. The latter is referred to as their ‘gold standard’ algorithm for estimating F.

An alternative approach to, or in support of, robust non-linear orthogonal regression algorithms is to attempt to remove the outliers from the data. Two competing methods are currently supported by the computer vision community. The Random Sample Consensus (RANSAC) algorithm, originally proposed by Fischler and Bolles [FB81] (reprinted [FB87]), randomly selects the minimum data set for the parameters required to compute the model (in this case the fundamental matrix) fit to the data and then computes the support for the postulated model across the whole data set. This method partitions data into outliers and inliers and eventually selects the minimum data set with the maximum support. The alternative is the Least Median of Squares (LMS) estimator which selects the model with the least median distance to all the data in the problem space from a limited selection of models computed from the minimum data set required for that model.

The above methods have been reviewed here in order to give an overview of the subject. Two excellent papers that review and compare all these methods are by Torr and Murray [TM97] and Zhang [Zha98]. While these methods are recommended in the referenced texts they are, in general, applicable to arbitrary stereo views and are, especially in the case of the latter examples, more complex than is required for periscopic stereo. Since the relative camera geometry is already known, the requirement in the case of periscopic stereo is for improved estimates of the internal camera parameters while minimizing the projected error of the corresponding points, subject to the constraints imposed by the epipolar geometry. The next section reviews a technique which performs this task.

6.2.1 The Variational Principle and Epipolar Calibration

The epipolar calibration method used in `Tina` is based on the *Variational Principle* proposed by Trivedi [Tri87]. Trivedi's technique for estimating the fundamental matrix was presented as a more stable alternative to the SVD solution prior to its use with data normalization. It requires no external variables or use of heuristics and is compatible with the generation of a covariance matrix that models and propagates the error in the camera's internal parameters and relative motion. The use of the covariance matrix is covered in Section 6.3. The following is a brief review of the method, as implemented by Thacker and Mayhew [TM91], together with the modifications necessary for use with periscopic stereo.

The basic idea of the variational principle is to obtain the smallest shift, or variation, in the observed data necessary to make that data fit the prescribed model which is defined by a set of parameters. To obtain the minimum shifts $\delta \boldsymbol{x}_j$ and $\delta \boldsymbol{x}'_j$ required to make the corresponding image points consistent with an estimate of the fundamental matrix F_j, [TM91] makes use of the Lagrange method of optimization by minimizing $\mathcal{E}$, the error, subject to:

$$\mathcal{E} = \sum_{j=1}^{n} \left(\delta \boldsymbol{x}_j^T \mathrm{S}^{-1} \delta \boldsymbol{x}_j + \delta \boldsymbol{x}'^T_j \boldsymbol{S}^{-1} \delta \boldsymbol{x}'_j \right) + \sum_{j=1}^{n} \lambda_j \left(\mathrm{F}_j + \nabla \mathrm{F}_j \delta \boldsymbol{x}_j + \nabla' \mathrm{F}_j \delta \boldsymbol{x}'_j \right) \tag{6.19}$$

where ∇ is the vector differential operator and S is the error given as a diagonal matrix with the components σ_x^2 and σ_y^2 for the image plane x and y directions and σ_z^2 which is set to zero, since the error is defined in terms of the image plane.

Analytically this is achieved as follows. From,

$$\frac{\partial \mathcal{E}}{\partial \delta \boldsymbol{x}_j} = 2\, \delta \boldsymbol{x}_j^T \mathrm{S}^{-1} + \lambda_j \nabla \mathrm{F}_j = 0 \qquad \text{and} \qquad \frac{\partial \mathcal{E}}{\partial \delta \boldsymbol{x'}_j} = 2\, \delta \boldsymbol{x}'^T_j \mathrm{S}^{-1} + \lambda_j \nabla' \mathrm{F}_j = 0 \tag{6.20}$$

rearrange to yield:

$$\delta \boldsymbol{x}_j = \frac{-\lambda_j \mathrm{S} \nabla \mathrm{F}_j^T}{2} \qquad \text{and} \qquad \delta \boldsymbol{x'}_j = \frac{-\lambda_j \mathrm{S} \nabla' \mathrm{F}_j^T}{2} \tag{6.21}$$

Expanding the epipolar constraint equation, $\delta \boldsymbol{x}'^{T}_{j}\, \mathrm{F}_j \delta \boldsymbol{x}_j = 0$, about the point $\boldsymbol{x}_j$, for Equation 6.19:

$$2\mathrm{F}_j + \nabla \mathrm{F} \lambda_j \mathrm{S} \nabla \mathrm{F}^T + \nabla' \mathrm{F} \lambda_j \mathrm{S} \nabla' \mathrm{F}^T = 0 \tag{6.22}$$

rearrange to yield:

$$\frac{\lambda_j}{2} = \frac{\mathrm{F_j}}{\nabla \mathrm{F}_j \lambda_j \mathrm{S} \nabla \mathrm{F}_j^T + \nabla' \mathrm{F}_j \lambda_j \mathrm{S} \nabla' \mathrm{F}_j^T} \tag{6.23}$$

Substituting in Equations 6.21,

$$\delta \boldsymbol{x}_j = \frac{\mathrm{F}_j \mathrm{S} \nabla \mathrm{F}_j^T}{\nabla \mathrm{F}_j \lambda_j \mathrm{S} \nabla \mathrm{F}_j^T + \nabla' \mathrm{F}_j \lambda_j \mathrm{S} \nabla' \mathrm{F}_j^T} \qquad \text{and} \qquad \delta \boldsymbol{x'}_j = \frac{\mathrm{F_j}}{\nabla \mathrm{F}_j \lambda_j \mathrm{S} \nabla \mathrm{F}_j^T + \nabla' \mathrm{F}_j \lambda_j \mathrm{S} \nabla' \mathrm{F}_j^T} \tag{6.24}$$

where

$$\mathrm{F}_j = \boldsymbol{x'}^{T}_{j}\, [\boldsymbol{t}]_\times \mathrm{R}\, \boldsymbol{x}_j \qquad \nabla \mathrm{F}_j = \boldsymbol{x'}^{T}_{j}\, [\boldsymbol{t}]_\times \mathrm{R} \qquad \text{and} \qquad \nabla' \mathrm{F}_j = \big[[\boldsymbol{t}]_\times \mathrm{R}\big]^T \boldsymbol{x}_j \tag{6.25}$$

Effectively the data shifts are created by variations in the estimates of the fundamental matrix created by optimizing the relative camera parameters.

The implementation given in `Tina` uses simplex optimization, minimizing the sum of the squares of error in the corresponding image points, subject to the epipolar constraint, as given in Equation 6.19. The camera parameters passed for optimization can be any of the internal camera parameters and the six parameters defining the relative camera transformation. Three for the translation and three for the rotation given in quaternion form[6], see Appendix F. In fact there are only five free parameters since,

$$q_0^2 = 1 - q_1^2 - q_2^2 - q_3^2 \qquad \text{and} \qquad t_1^2 = 1 - t_2^2 - t_3^2 \tag{6.26}$$

The implementation in `Tina` was originally written for a stereo camera system with two real cameras and therefore two sets of internal camera parameters. However, by including a new "register" (temporary result storage for optimization process) function and a conditional statement the

[6]A Quaternion is a four component vector, originally defined by Hamilton 1843 [Ham66], which can be used to encode 3D rotations.

source code has been modified to optimize a single set of camera parameters, as shown in Figures 6.3 and 6.4. The original error function, `triv_camerror(...)`, computes an estimate of the fun-

```
/* cam_error.c */ .......
double  triv_camerror(int *n_data, double *x,
                      Camera * cam1, Camera * cam2, List * world3d,
                      Vec2 * (*pix_get1) ( /* ??? */ ),
                      Vec2 * (*pix_get2) ( /* ??? */ ), double accuracy)
{...../* original Tina source code */
}

double  scam_stereo_reg(Covar * incov, int mask, double *a)
{ /* new stereo_reg for single_camera stereo. used in
   * pixchisq_scam() for cam_cal_triv_simplex()., Added May 2001, by Ed.
   */
    Matrix *delta, *dprod;
    double  chisq = 0.0;
    int     i, n_par = 0;

    if (incov == NULL)  return (0.0);

    for (i = 0; i < 16; i++) if (mask & (1 << i)) n_par++;

    delta = matrix_alloc(1, n_par + 6, matrix_full, double_v);
    for (i = 0; i < n_par + 6; i++) {
      delta->el.double_v[0][i] = a[i] - VECTOR_DOUBLE(incov->vec, i);
    }
    dprod = matrix_prod(delta, incov->mat);
    for (i = 0; i < n_par + 6; i++) {
      chisq += dprod->el.double_v[0][i] * delta->el.double_v[0][i];
    }
    matrix_free(delta);  matrix_free(dprod);
    return (chisq);
} .......
```

Figure 6.3: Modified source code for error storage function for epipolar calibration with a single set of camera parameters

damental matrix and sum of square error of the projected corresponding points for every iteration of the simplex optimization algorithm. The new register function, `scam_stereo_reg(...)`, updates the covariance error matrix and the estimates of the camera parameters from the previous covariance error matrix. The source code for the simplex optimization algorithm is unmodified and not included here.

The main advantage of this method of epipolar calibration is that it is simple, fast and the accuracy of the estimates reportedly [Tri87, TM91] improves over time. This is achieved by the use of error

```
static double pixchisq_scam(int n_par, double *a)
{ /* New function */
  double  chisq = MAXDOUBLE,  *f = NULL;
  (void) store_camera_int(cal_mask, a, cal_caml);
  (void) store_camera_int(cal_mask, a, cal_camr);

  if (store_camera_rel(a + n_par - 6, cal_caml, cal_camr))
    {
      int     n = MAXINT;
      chisq = triv_camerror(&n, f, cal_caml, cal_camr, cal_data,
                            cal_get_pixl, cal_get_pixr, accuracy);
      chisq += scam_stereo_reg(cal_in_cov, cal_mask, a);
    }
  return (chisq);
}
static double pixchisq_stcam(int n_par, double *a)
{  ............/*** AS ABOVE ***/
  (void) store_camera_int(cal_mask, a, cal_caml);
  (void) store_camera_int(cal_mask, a + n_par / 2 + 3, cal_camr);
  if (store_camera_rel(a + n_par / 2 - 3, cal_caml, cal_camr)) {
      int     n = MAXINT;
      chisq = triv_camerror(&n, f, cal_caml, cal_camr, cal_data,
                            cal_get_pixl, cal_get_pixr, accuracy);
      chisq += stereo_reg(cal_in_cov, cal_mask, a);
    }
  return (chisq);
}

double  cam_cal_triv_simplex(Camera *caml, Camera *camr, int mask, Bool single_cam,
                 List *data, Vec2 *(*getpixl)( /*???*/ ), Vec2 *(*getpixr)( /*???*/ ),
                 Covar *inv_cov) /* inverse covarience */
{
    double  *a, chisq, chisq_old;
    double  (*pixchisq)( );
    int     n_par, n_parms, i;
    if (data == NULL || caml == NULL || camr == NULL) return (0.0);
    cal_mask = mask; cal_data = data; cal_caml = caml; cal_camr = camr;
    cal_get_pixl = getpixl; cal_get_pixr = getpixr; cal_in_cov = inv_cov;
    for (i = 0, n_par = 0; i < 16; i++) if (mask & (1 << i)) n_par++;

    if ( single_cam ) {         /***** MODIFIED ******/
      n_parms = n_par + 6;
      pixchisq = pixchisq_scam;
    }
    else {
      n_parms = 2 * n_par + 6;
      pixchisq = pixchisq_stcam;
    }
    a = (double *) ralloc((unsigned) n_parms * sizeof(double));
    (void) conv_camera_int(mask, caml, a);
    (void) conv_camera_rel(caml, camr, a + n_par);
    if ( !single_cam )
      (void) conv_camera_int(mask, camr, a + n_par + 6);

    chisq_old = pixchisq(n_parms, a);
    for (i = 0; i < 5; ++i) {

       chisq = simplexmin(n_parms, a, scale_init, pixchisq, c_test1,
                          (void (*) ()) format);
       if (chisq_old - chisq < c_test2)
           break;
       chisq_old = chisq;
    }
    (void) store_camera_int(mask, a, caml);
    (void) store_camera_rel(a + n_par, caml, camr);
    if ( !single_cam )
      (void) store_camera_int(mask, a + n_par + 6, camr);
    else
      (void) store_camera_int(mask, a, camr);
    rfree((void *) a);
    return (chisq);
}......
```

Figure 6.4: Modified source code for optimization of a single set of camera parameters, subject to the epipolar constraint

covariance and is discussed in Section 6.3. An initial estimate of the error covariance is required by the implementation and is derived from a weak model derived from the initial guess values of the chosen internal and relative camera parameters. The implementation also includes error bounds which limit image plane errors, in pixel units, and ensures a slow and steady convergence to the desired result.

Section 6.1.3 identified that the error in the location of the principle point was considerable for the particular camera used in these experiments. This leads to the question of whether the variational principle for epipolar calibration can reduce this initial error and if so how many iterations of the optimization process are required. In order to answer these questions an experiment similar to that in Section 6.1.3 was conducted.

Beginning at the centre of the image, (192, 144), a number of initial calibrations were computed, each time initializing the covariance matrix so that the result in always independent. Table 6.3 shows the results using the two left hand images in Figure 6.1. The corresponding image data was constructed using the labelled points on the calibration grid which are, in general, accurate to less than a pixel. It should be noted that only the corresponding image points are used, not the world data points. The use of this data is discussed further in Section 6.4. An initial estimate of the relative camera transformation, calculated from Equation 3.4 in Chapter 3, was included via the `Tina's` camera parameters window[7]. The initial value of focal length was $f_0 = 10.0$ and the horizontal scaling was $a_x = 1.1$ in each case.

Table 6.3 is arranged with the initial estimates of the coordinates of the principle point (u_0, v_0) in the far left columns and the results of the optimized internal camera parameters in the right hand columns. $\sum \varepsilon^2 \, Epi_{min}$ is the sum of the square error given by Equation 6.19 for all the calibration points. The results in Table 6.3 demonstrate several things. Firstly, the estimated value of the horizontal scaling is erratic and yields illogical results (labelled '‡') in cases where the initial principle point

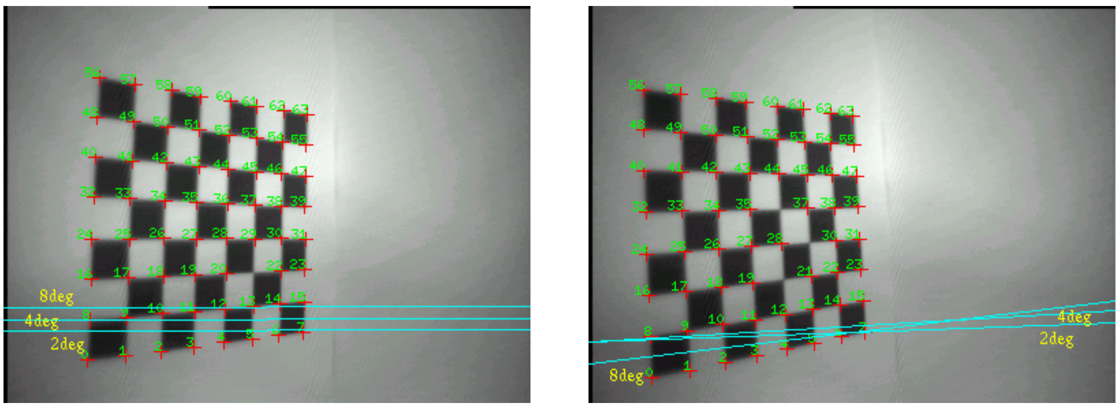
[7]The calibration tool in `Tina` effectively allows any of the camera parameters to be initialized prior to calibration

	Guess P		$Epi_{min}(f,\ u_0,\ v_0,\ a_x)$					
start	u_0	v_0	$\sum \varepsilon^2\ Epi_{min}$	u_0	v_0	a_x	f_{min}	*end*
	176	139	0.6169	176.0	139.0	1.116	10.01	?
	184	139	0.6138	184.0	139.0	1.082	10.00	?
	192	139	0.6791	192.0	139.0	‡ -1.92	10.00	
	200	139	0.6893	199.9	138.9	‡ -1.96	9.99	
	184	144	0.6204	184.0	144.0	1.354	9.99	
#	192	144	0.7687	192.0	144.0	‡ -2.23	10.00	
	200	144	0.7880	200.0	144.0	‡ -2.31	9.99	
	184	149	0.7924	184.0	149.0	‡ -2.29	9.98	
	192	149	0.7489	192.0	149.0	‡ -2.19	9.98	
	200	149	0.7027	200.0	149.0	‡ -1.861	9.96	

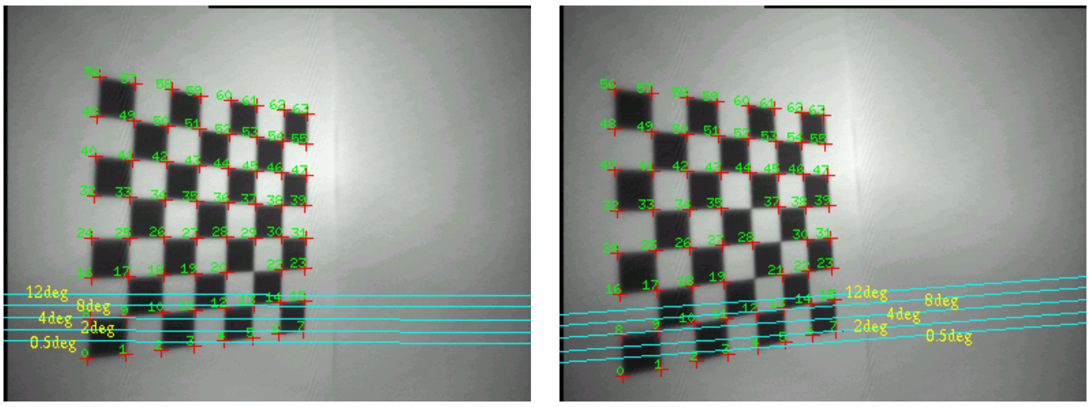
Table 6.3: Initial results for epipolar calibration with varying estimates of the principal point.

is grossly inaccurate. However, when the initial estimate of the principle point is closer to the true value more sensible results are achieved and the sum of the square error is reduced. Secondly, there is little change in the optimized estimates of the principle point or the focal length. This situation is maintained even when the horizontal scaling is removed from the parameter list for optimization. This is not surprising since a reasonably accurate initial estimate of the relative transformation is supplied in the parameter list for optimization. With only small changes in the relative camera parameters there is little opportunity for the simplex algorithm to optimize the coordinates of the principle point. This is further compounded by two facts. Firstly, the internal camera parameters are the same for both cameras, as determined by the modification for periscopic stereo use. Second, the principle point creates an relative shift of the data point and not a scale change. This is evident from the camera to image transformation given by Equations A.5 and A.6 of Appendix A. Therefore any change in estimates of the principle point will have less effect than that of the horizontal scaling of focal length on the projected image points.

Although the variational principle can not aid the search for better estimates of the coordinates of the principle point, its use for recovering the relative camera geometry and therefore maintaining the calibration accuracy of an already calibrated system is evident. Figures 6.5(a) and 6.5(b) show the effect of deliberately changing the initial estimate of the relative camera transformation by varying the incremental rotation angle of the periscopic head from the standard $4°$ to $2°$ and $8°$. Figure 6.5(a)



(a) before epipolar calibration



(b) after epipolar calibration

Figure 6.5: The effect of varying the relative camera transformation on epipolar calibration.

shows the effect on the epipolar lines before calibration and Figure 6.5(b) after. No visible change is apparent in latter, even though the variation in the relative transformation between the virtual cameras is far greater than would ever be apparent in a real system, excluding catastrophic failure. The difference in the sum of the square error was less than 0.01 of the figure for the (192,144) entry in Table 6.3, which was the principle point of the default parameters used for this particular test.

The ability of the variational principle to maintain the accuracy of calibration over an extended period of time far outweighs the inability to locate the principle point. The reason for including the apparently negative result, of the latter, is explained further in Section 6.4.

6.3 Maintaining Calibration and Propagating Error

All of the camera calibration techniques mentioned in this chapter include some form of iterative optimization, either directly or as a recommendation for their use. The requirement for iterative optimized solutions implies a need to assess and monitor the accuracy of the estimated results. The error, or uncertainty, in such results can be given by the error covariance and modelled by the inverse covariance matrix. The computation of the error covariance also allows for the optimal combination of two estimates of the parameter sets that define the optimized function. An introduction to covariance estimation and optimal combination by Thacker and Cootes [TC96] can be found at *CVOnline*. Alternative treatments on the computation of uncertainty are given in [Fau93] and [HZ00]. A detailed description of the computation of error covariance is not included and the following is given without qualification.

Given an error metric χ^2 from an optimization cost function $f(\boldsymbol{a})$ for a set of parameters $\boldsymbol{a}$, the expected change in χ^2, or error covariance, for a small change in model parameters is given by:

$$\Delta\chi^2 = \Delta\boldsymbol{a}^T\mathcal{C}^{-1}\Delta\boldsymbol{a} \tag{6.27}$$

where $\mathcal{C}^{-1}$ is the inverse covariance matrix. This is based on the form given in [TC96].

The inverse covariance matrix is computed from the partial derivative, or Jacobian, matrix $\delta f/\delta\boldsymbol{a}$ of the original cost function used in the optimization of the parameters. The computation of the error covariance is implemented in `Tina` using the Jacobian matrix technique. This is used to derive an estimate of the error and fed back into the optimization processes used in Section 6.2.1 and mentioned later in Section 6.4. In keeping with the modification to the source for epipolar calibration described in Section 6.2.1, the source code for the computation of the covariance matrix has been modified for a single set of internal camera parameters in a stereo system. The modification is almost identical, in practice, to that given in Section 6.2.1 and is therefore not included here.

The use of the error covariance in the optimization loop effectively allows previous estimates of the parameters to be combined with the new estimates. This propagates the error and produces a robust maintenance of the calibration accuracy over time. The requirement for a robust epipolar calibration algorithm is particularly useful with Trivedi's variational technique, since the method does not incorporate any form of outlier detection or removal. Any outliers in the correspondence data are incorporated in the optimization and can lead to the degradation of the calibration accuracy with each subsequent optimization. The inclusion of the error covariance estimate tends to reduce the effect of outliers in the data over time, assuming their occurrence and effect have a uniform distribution. However, a large percentage of gross outliers in the correspondence data would have an instant and longer lasting effect. Such an occurrence is unlikely during the initial calibration since the possibility of outliers in the data set is low (not withstanding catastrophic failure of the grid labelling algorithm) when using the calibration grid. However, for the continued re-calibration of the system, using naturally occurring correspondence data from the image scene, the probability of the existence of gross outliers increases. The performance of correspondence algorithm depends of the epipolar constraint and the re-calibration of the stereo cameras, defines the epipolar geometry. The two processes are therefore inextricably linked.

Chapter 5 included some simple experiments comparing the performance of correspondence algorithms applied to both rotationally corrected and uncorrected stereo images. In Chapter 3 the analysis of the relationship between the image data and the rotating mirror identified that the image data is subjected to rotation in sympathy with the mirror and assumed that the centre of rotation is coincident with the optical axis. It was stated that the experiments with the uncorrected images, in Section 5.3, allowed for the simulation of degraded calibration accuracy and examined the effect on correspondence matching. The reason for this degradation in calibration accuracy is due to the fact that rotation about an axis through the image plane which is not, in fact, coincident with the optical axis of the camera. This translates to an error between successive relative camera transformations which accumulates over half a cycle of a complete scan by the periscopic head. The error then decreases over the other half cycle, returning the relative camera transformation to the original state. This effectively induces an oscillation, relative to each camera, about the horizontal centre line of the image, such that there is a variable, vertical shift between any selected stereo pair. Figure 6.6 shows an image pair ten frames ahead of the currently calibrated pair. The epipolar line, computed from the current calibration, clearly demonstrates the vertical shift induced by the displacement (compare both ends of the epipolar line and objects in the scene). This is discussed further is Section 6.3.1.

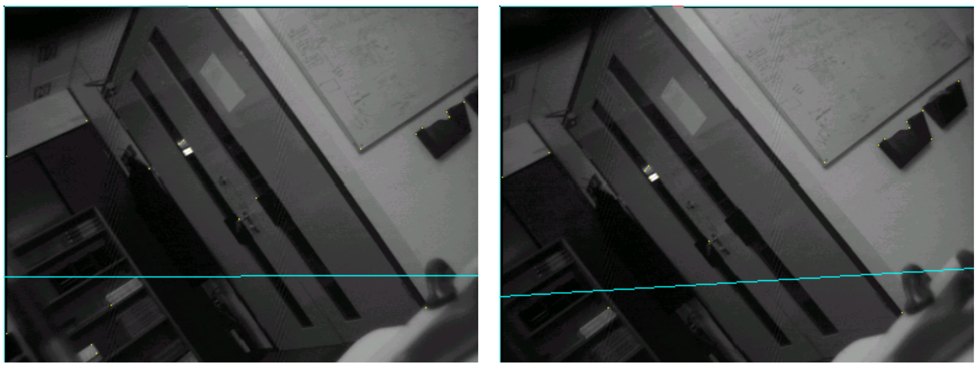
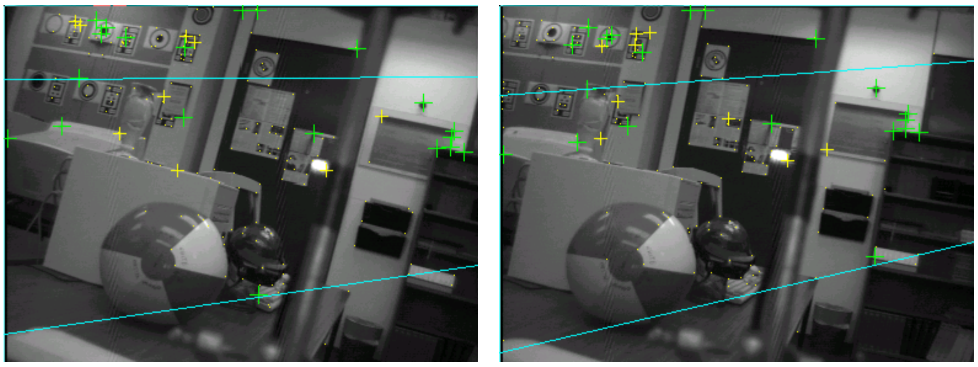

(a) left image (b) right image

Figure 6.6: Vertical shift induced by the displacement of the axis of rotation from the optical axis.

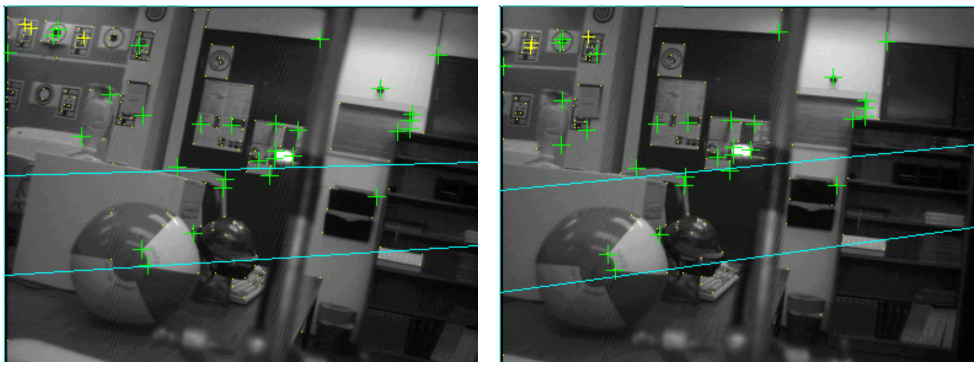
In order to assess the ability of the epipolar calibration, using the variational principle, to maintain calibration accuracy over time and also the effects of this vertical shift error, the following experiment was conducted. Starting with a pre-calibrated system, as described in Section 6.4, an image pair was selected from a periscopic sequence, approximately one frame prior to the currently calibrated alignment[8]. The epipolar calibration was computed for the pair using the corresponding image points extracted from the scene as shown in Figures 6.7 and 6.8. Four internal camera parameters, (f, u_0, v_0, a_x), are included in the optimization. The error covariance was then computed, combined with current covariance and saved for the next iteration together with the new estimate of the calibrated parameters for the camera models. The sequence was advanced by one frame (old right hand image becomes new left hand image) and the corresponding points computed using the rotated patch algorithm with the simulated image plane rectification, as described in Section 5.2. The rectification is derived from the last estimate of the epipolar geometry. The epipolar calibration is computed using the new correspondence data and the whole cycle repeated until insufficient data points are recovered to compute the calibration.

Figure 6.7(d) shows the images where the calibration is most accurate. This is due to the similar relative image alignment with the image frames from the grid calibration. The epipolar lines shown indicate the alignment of corresponding structure in the images. Figure 6.7 shows the first half of the image sequence with frames before this starting point and Figure 6.8 the second half of the image sequence with frames after this point. The results of this "calibration tracking" experiment are given in Table 6.4 and the order in which each image pair was process is indicated. The reason for selecting images pairs in the forward and back sequence is to maintain the calibration accuracy as long as possible for the purposes of demonstrating the effects which are the subject of interest in this experiment. The horizontal and vertical disparity, given in the second column, are computed from the estimated camera centre coordinates subject the rectification matrix in the parallel camera model (see Appendix E). F_{min} is the lower of the total number of features in each image and the epipolar band

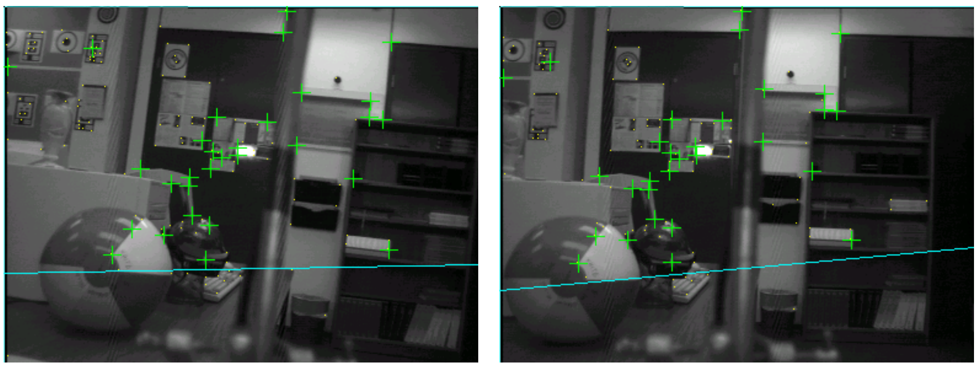
[8]There is no correlation in image frame numbers when switching from the calibration image sequence to a sequence of the imaged scene. Therefore the geometric alignment between frames of different sequence is only approximate



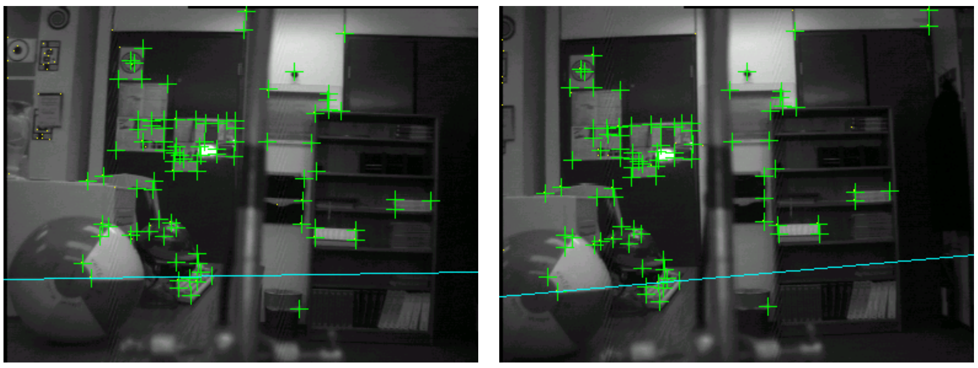
(a) frames 20:21



(b) frames 21:22

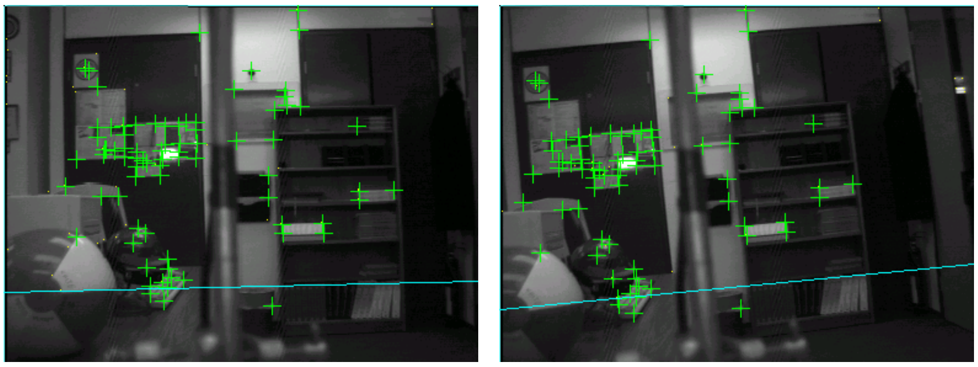


(c) frames 22:23

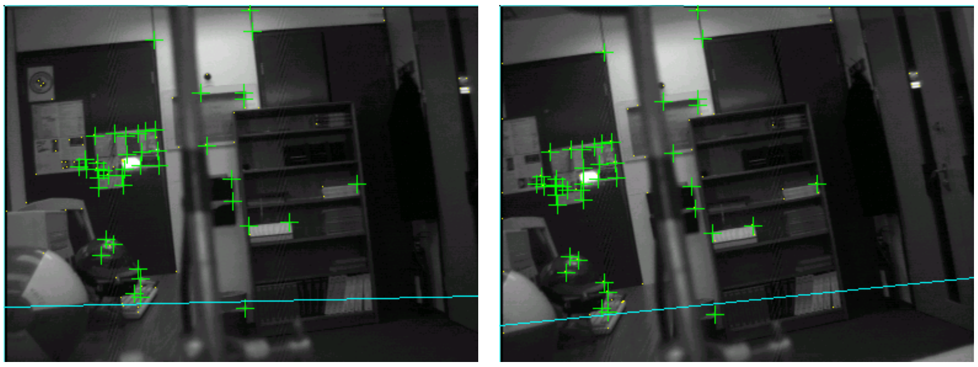


(d) frames 23:24

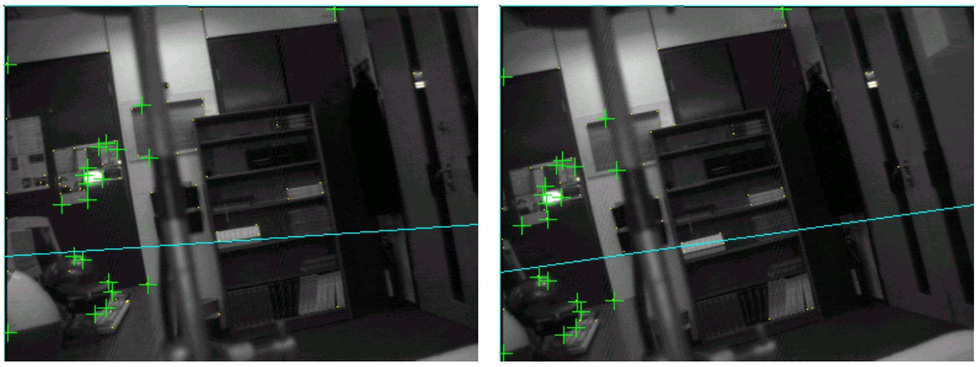
Figure 6.7: Results from calibration tracking experiment - part 1; (a) frames 20:21, (b) frames 21:22,



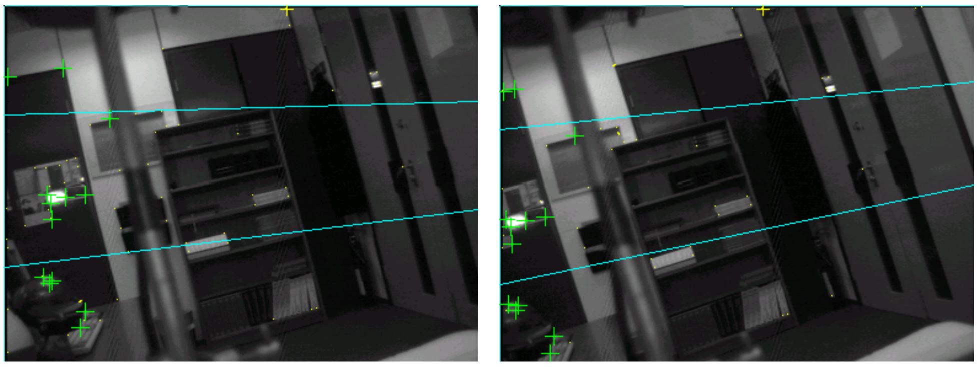
(a) frames 24:25



(b) frames 25:26



(c) frames 26:27



(d) frames 27:28

Figure 6.8: Results from calibration tracking experiment - part 2; (a) frames 24:25, (b) frames 25:26,

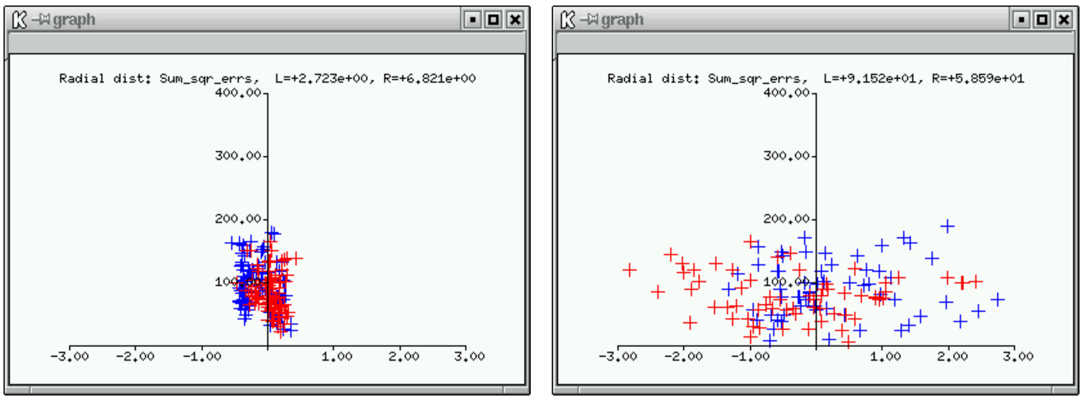
order	*frames (l:r)*	*disparity (h,v)*	F_{min}	*e-band*	*matched*	$\sum \varepsilon^2\ Epi_{min}$	$\sum \varepsilon^2\ Rad_{dst}(l,r)$
7	20:21	20.62, -1.53	166	6	20	89.74	1965, 272.1
5	21:22	18.80, -1.71	139	5	30	197.8	750.6, 89.63
3	22:23	17.68, -2.56	119	3	72	78.50	23.50, 89.63
1	23:24	17.57, -2.43	106	3	89	10.29	5.62, 31.48
2	24:25	17.72, -2.24	96	3	78	57.31	26.49, 9.11
4	25:26	17.80, -2.34	82	4	33	168.4	62.0, 323.9
6	26:27	19.23, -1.23	69	5	21	139.3	803.4, 1625
8	27:28	20.62, -1.53	53	5	13	123.6	5430, 409.4

Table 6.4: Results of calibration tracking in the presence of vertical shift error.

width (labelled '*e-band*') is the width, in pixel units, of the search band about the epipolar lines, as described in Section 5.2. This was used in an attempt to compensate for the vertical shift error and allow the correspondence algorithm a greater opportunity to produce sufficient data for calibration. At the limits of the calibration sequence the width of epipolar band has already exceeded a sensible limit. Refer to the discussion on setting the width of the epipolar band on page 93 in Section 5.2 for the qualification of a "sensible limit". The sum of the squares error for the epipolar optimization, for the parameters ($f,\ u_0,\ v_0,\ a_x$), is the the same as that used in Table 6.3 from Equation 6.19. This gives a measure of the total error of the corresponding points perpendicular to the respective epipolar lines. This gives a reasonable guide but not a particularly accurate measure of the overall accuracy of the calibration since it does account for horizontal error due to changes in the location of the principal point ($u_0,\ v_0$) and horizontal scaling factor (a_x) during the successive optimization. Although no actual lens distortion parameters are included in the model a measure of the radial lens distortion (labelled '$\sum \varepsilon^2\ Rad_{dst}(l,r)$') is included in Table 6.4 to give a better overall indication of accuracy. This error measure is computed between the corresponding image and world ($\boldsymbol{X}$) point as:

$$\sum \varepsilon^2\ Rad_{dst} = \sum \left((X_u^2 - X_v^2)^{\frac{1}{2}} + (x_u^2 - x_v^2)^{\frac{1}{2}} \right)^2 \qquad (6.28)$$

on the image plane (u, v) where the world points are back-projected from the images points via the camera models and the computed disparity and then re-projected back on the image plane. This may seem convoluted but the image plane is the only place where accurate measurement can be made and the double projection ensure all the model parameterized are included in the error measure. It should be noted that the figures do not give an absolute measure in themselves since they are the sum of the squared error over all the data points. The best measure of the calibration accuracy is the distribution of the the radial lens errors, as shown in Figure 6.9, together with the computed values of both the epipolar and radial distortion error measures. Since there is a limit to the amount of data, especially



(a) accurate calibration

(b) weak calibration

Figure 6.9: Examples of distributions of radial lens distortion errors and their indication of calibration accuracy.

across multiple images of error distribution patterns, the reader is requested to use the same subjective assessment as the author. That is, if both error measures are low in value and the left and right radial distortions are similar then the calibration accuracy is high. If the figures rise or become unbalanced then the accuracy is falling.

The increase in vertical shift error is evident from both the disparity figures in the second column of Table 6.4 and Figures 6.7(a) and 6.8(d). Although the numerical difference in vertical disparity appears small it should be noted that these are for the principle point near the centre of the image. The relative distortion created by the axial offset between the centre of rotation and the optical axis is not uniform across the image. Apart from the induced vertical shift there is also an equivalent yaw error component which equates to an extra rotation in the image data. This is evident from the fluctuation in the horizontal disparity and by comparing the displacement in the corresponding structure about the epipolar lines in Figures 6.7(a) and 6.8(d). The displacement is greater at the edges of the image than it is in the centre. The reason for this is described in Section 6.3.1.

Although the algorithm makes a valiant attempt (the sum of the squares error for the optimization is small in global terms) to track the change in the relative camera transformation it is incapable of maintaining calibration accuracy for more than a few frames. In Section 6.2.1 the ability of the variational principle to estimate the relative camera transformation was found to be rather good, even in the presence of a large difference between the initial estimate and the apparent, imaged, geometry. However, the accuracy of the correspondence data was extremely good, since it was derived from the calibration grid. Here, the correspondence data contains both localization noise and gross outliers. Some of these are induced by an attribute of experiment which, in reality, would not exist; namely the camera's support stand which appears in the centre of the image. This creates invalid features from points of occlusion. Given the limitations of the experimental apparatus and the fact that the change in the relative camera transformation is severe the performance of the algorithm is not as bad as the results suggest.

Maintaining calibration accuracy autonomously could be achieved by keeping track of the percentage of data points recovered using the measure, or something similar, discussed in Chapter 5. If the number of "fixed" correspondence matches compared to the potential number correspondence matches falls below a certain threshold, a re-calibration could be initiated. The requirement to re-calibrate for every successive frame is not envisaged for rotationally uncorrected image data from a real periscopic system and certainly not necessary for rotationally corrected data as demonstrated in Figure 5.4 in Chapter 5.

6.3.1 Modelling the Axial Error

The large offset error between the centre of rotation and the camera's optical axis identified in Section 6.3 would not exist in an accurately manufactured periscopic stereo head. This type of error is extreme and only induced by the rotationally uncorrected, captured from the turn table implementation of periscopic stereo described in Section 3.4. It should be noted however, that while the rotational correction eliminates this error by interpolating across the whole image, it does so at the expense of localization accuracy of the corresponding data. The offset error is effectively transferred to positional error of the imaged structure and therefore would be incorporated in the reconstruction of the scene.

A model of the system and the axial offset can be created using the analysis in Section 3.1. By replacing the zero x_m and z_m components of $\boldsymbol{C}_r$ in Equation 3.4 with δu and δv (for a displacement of the image plane) yields:

$$\boldsymbol{C}_v = \begin{bmatrix} \delta u \\ b \\ \delta v \end{bmatrix} - 2b\cos\theta \begin{bmatrix} \sin\theta\, \sin\phi \\ \cos\theta \\ \sin\theta\, \cos\phi \end{bmatrix} = \begin{bmatrix} \delta u - b\sin 2\theta\, \sin\phi \\ -b\cos 2\theta \\ \delta v - b\sin 2\theta\, \cos\phi \end{bmatrix} \tag{6.29}$$

Introducing the same components to Equations 3.5, 3.8, 3.9 and repeating the derivation, a new version of the transformation matrix in Equation 3.13 is derived, where:

$$\mathrm{T}_{(\delta u,\delta v)} = \begin{bmatrix} 1+\frac{2\sin^2\theta\, \sin^2\phi\delta x\delta u}{(1-\delta x)\delta u} & -\sin 2\theta\sin\phi & \frac{2\sin^2\theta\, \sin\phi\, \cos\phi\delta z\delta v}{(1-\delta z)\delta v} \\ \frac{\sin 2\theta\, \sin\phi\, \delta x\delta u}{(1-\delta x)\delta u} & (\cos 2\theta) & \frac{\sin 2\theta\, \cos\phi\delta z\delta v}{(1-\delta z)\delta v} \\ \frac{2\sin^2\theta\, \sin\phi\, \cos\phi\delta x\delta u}{(1-\delta x)\delta u} & -\sin 2\theta\cos\phi & 1+\frac{2\sin^2\theta\, \cos^2\phi\delta z\delta v}{(1-\delta z)\delta v} \end{bmatrix} \tag{6.30}$$

The denominators in the first and third columns of Equation 6.30 demonstrate the compound nature of the distortion induced by an axial offset.

Selecting the assumed principle point $\boldsymbol{p}_o = [\,imagewidth/2,\ imageheight/2\,]$ and an arbitrary image point toward the periphery of the frame $\boldsymbol{x}_i = [\,3\,imagewidth/4,\ 3\,imageheight/4\,]$ and converting to the mirror coordinate frame yields $\boldsymbol{p}_m = [\,0,\ b,\ 0\,]$ and $\boldsymbol{x}_m = [\,x_u\,pix,\ b,\ x_v\,pix\,]$ where pix is the assumed pixel size. Transforming these points with Equation 3.13 to the virtual image plane creates a reference line defined by $\boldsymbol{P}_{vr}$ and $\boldsymbol{X}_{vr}$. Note that the new subscript "vr" defines "virtual" and is different from the nomenclature used in Chapter 3.

Repeating this process for an offset principle point and the chosen image point, offset by the same translation, with the transformation matrix given in Equation 6.30 creates a second line on a distorted virtual image plane defined by $\boldsymbol{P}'_{vr}$ and $\boldsymbol{X}'_{vr}$. The absolute difference between these points and the reference is given by:

$$\|\boldsymbol{X}'_{vr} - \boldsymbol{X}_{vr}\| \qquad \text{and} \qquad \|\boldsymbol{P}'_{vr} - \boldsymbol{P}_{vr}\| \tag{6.31}$$

and effectively yields a point measure of the image plane error, in pixels, induced by the axial offset. Assuming perfect calibration and the rotational error witnessed in Figures 6.7(a) and 6.8(d) is approximated by the absolute difference given in Equation 6.31, then a rough estimate of the maximum axial offset (2D translation of the image plane) was found by iterative testing until the error in the localization of the peripheral point falls below the sensible limits of the epipolar band given in Section 5.2, across two frames separated by the $\phi = 4°$.

While this is not a particularly accurate model of the system, an estimate of the maximum allowable axial offset was found to be approximately ± 8 pixels from the central position. The calibrated principle point for the camera used in these experiments was found to be approximately (162, 138), which is acceptable in vertical direction but not in the horizontal. Although such errors are common in low-cost CCD cameras higher, precise localization of the principle point is certainly possible with modern manufacturing and factory calibration techniques. It is recommended that a camera with a principle point of with a maximum limit of ± 5 pixels be used in a periscopic stereo head. The camera's optical axis can be aligned to the axis of rotation by using a calibrated jig during manufacture.

6.4 Combining Grid and Epipolar Calibration

The rationale behind the experiments in Sections 6.1.3, 6.2.1 and 6.3 is that the calibration process should, ideally, be an autonomous process. It can be argued that this is desirable for any camera system used for 3D reconstruction. However, the target applications of remote operation and large-scale reconstruction requires that an automated calibration, or at least re-calibration, scheme is mandatory.

The concept of "calibration in a box" was introduced in Section 6.1.2 and essentially allows for the possibility of applying grid calibration to the periscopic stereo system prior to use. However, the use of the grid data is not exclusive. Assuming the failure modes (when block mismatches occur, as shown in Figure 6.2) are identified and defensive techniques applied, the set of corresponding points from the imaged calibration grid, is very accurate. It is therefore logical to make use of this information for initial epipolar calibration, as shown in Section 6.2.1.

In Sections 6.1.3 and 6.2.1 limitations are identified with both the currently available techniques in deriving the initial calibration of an unknown stereo camera system. Without prior knowledge of the internal camera parameters, the initial calibration is laborious process consisting of the iterative estimation of the parameters and gradual modification of the camera model. While other techniques may improve the situation and ultimately lead to a more autonomous calibration procedure, there still remains the question of combining these two calibration techniques in order to achieve a full calibration model for the system. Epipolar calibration can not recover absolute scale and grid calibration, on it's own, can not recover the epipolar geometry of a stereo system. The latter is the most important constraint applied to both the acquisition of correspondence data and subsequent 3D reconstruction.

Initially it was thought, by the author of this dissertation, that the "optimal" solution to stereo camera calibration was to calibrate for absolute scale first, using a grid calibration method. Then improve the estimates of the camera's internal parameters and derive relative transformation from updated estimates of the fundamental matrix acquired by epipolar calibration. This idea is suggested in [JKS95] and implemented by Zhang *et al*in [ZFD97]. The sequence of operations for the method given in [ZFD97] is as follows[9]:

- Point the cameras to toward the calibration apparatus and match the corresponding world and image points.
- Point the cameras toward the environment, extract points of high curvature in both images and perform robust correspondence matching of these points.
- Estimate the epipolar geometry by using the two sets of matches (reference + environment).
- Reconstruct in a projective basis from the points of reference.
- Estimate the projective distortion in order to recover structure in Euclidean space.

The problem with this approach is that if the initial grid calibration is not particularly accurate, the subsequent epipolar calibration may, depending on the technique employed, require a number of iterations before sufficiently accurate estimates of the internal camera parameters are achieved. Furthermore, any error in the absolute scale of the world to camera transformation can not be improved by epipolar calibration. Zhang, *et al* [ZDFL95], recognizes that linear solutions for estimating the fundamental matrix are sensitive to noise in image point localization and from mismatches. This fact is supported in [HZ00, SHB99, Har95, TM97]. In [Zha98], Zhang explores in detail robust methods for

[9]An outline of Zhang's method is given on their web pages at:

`http://www-sop.inria.fr/robotvis/personnel/zzhang/CalibEnv/CalibEnv.html` estimating the fundamental matrix in the presence of noise, yet fails to make use of the more accurate data available from the calibration grid in the initial phase. The following method of calibration was derived for use with the concept of "calibration in a box". However, it is equally applicable to any stereo camera system that identifies and labels the calibration grid points in an ordered fashion.

Instead of initially calibrating to recover the external camera parameters and then recover the relative transformation between the cameras, the sequence is reversed and extra epipolar re-calibration performed at the end. A summary of the algorithm for "calibration in a box" is given in Figure 6.10 and the description proceeds as follows.

1. Initial Epipolar Calibration:
  Using the corresponding image points from a calibration grid, compute the epipolar calibration, assuming the left optical centre is coincident with the world origin. Optimize all the camera parameters for minimal epipolar error and save error covariance.

2. Grid Calibration:
  Compute the grid calibration for the left camera.
    a: If using Tsai's grid calibration optimize the initial estimate of the focal length only.
    b: Copy across the internal parameters and computer the external parameters for the right camera.
    c: Optimize all the parameters for minimal error between the projected calibration and corresponding image points.
    Combine the error covariance with previous estimate.

3. Final Epipolar Calibration:
  Repeat epipolar calibration and optimize all parameters.
  Computer final error covariance and store in case of re-calibration.

Figure 6.10: Summary of the Calibration in a box algorithm

Firstly, using the calibration grid described in Section 6.1.2, compute the epipolar calibration using the variational principle, as reviewed in Section 6.2.1. This yields an accurate estimate of the relative camera transformation. An initial estimate of the horizontal scaling factor is also achieved, assuming the initial guess of the coordinates of the principle point is not in gross error. The initial guess values for the internal camera parameters used for the calibration results achieved in this section were; $f =$

$10.0,\ u_0 = 161,\ v_0 = 139,\ a_x = 1.05$. The initial guess of the focal length is unimportant, as described in Section 6.1.1, but is necessary for the initialization of the covariance matrix for the desired number of internal camera parameters prior to the epipolar calibration, as described in Section 6.2.1. The initial values the external parameters are determined for right hand camera from the calculation of the relative transformation given by Equation 3.4. The optical centre of the left hand camera is initially assumed to be coincident with the origin of the world coordinate frame, as described in Section 6.2.1. After the epipolar calibration is computed, the covariance matrix is derived for the estimated internal and relative camera parameters and stored for later use.

The next stage is to compute the grid calibration to recover the external camera parameters and improve estimates of the internal camera parameters. This is achieved in three phases; Firstly, using Tsai's method with subsequent image plane optimization, as described in Section 6.1.1, an estimate for the focal length is derived. No other parameters are included in the optimization. An alternative, possibly more robust (see Section 6.1.4), grid calibration method could equally be applied at this point. The second phase addresses the fact that both the left and right virtual cameras in a periscopic stereo system have the same internal parameters. In a conventional stereo system the first stage would be conducted twice and following phase ignored.

There are two approaches which could be adopted. The focal length and the external camera parameters can be recovered for the second camera as above and the internal parameters combined with those from the left camera to yield an average. This should, ideally, reduce the overall error by averaging over both sets. However, the optimized estimates from Tsai's method are not that stable (as shown in Section 6.1.3) and the extra computational cost seems, in the opinion of the author of this dissertation, to outweigh the minimal advantage for such a naive combination at this time. An alternative grid calibration method may make this option more attractive.

Since the virtual cameras have the same internal parameters it seems unnecessary to compute them twice. The internal camera parameters can be simply copied across from the left to the right. However, the external camera parameters for the right hand camera are still required. This can be solved by using the following geometrical relations. With reference to Figure 6.11, given any two of the 3D transformations relating to a stereo camera system the third can be derived directly from the combination of the other two such that:

$$[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wl} = [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{c}^{-1}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wr} \quad \text{or} \quad [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{c} = [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wr}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wl}^{-1}$$
$$\text{or} \quad [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{c}^{-1} = [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wl}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wr}^{-1} \tag{6.32}$$

In this particular case we have the relative transformation $[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{c}$ and the left world to camera

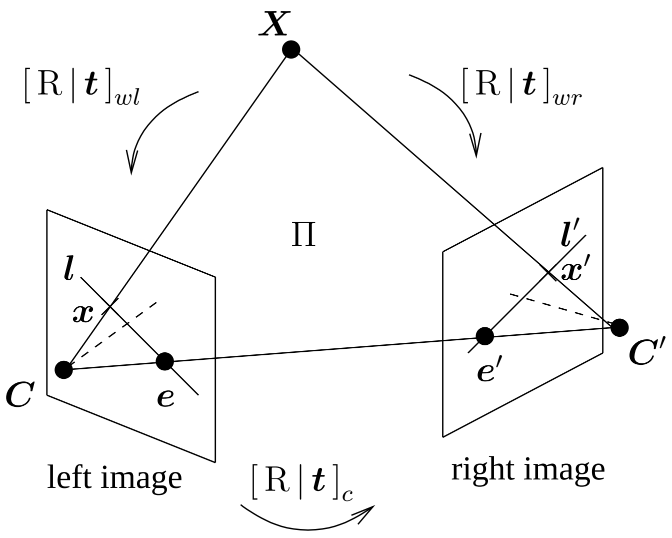


Figure 6.11: Relating the transformation in epipolar geometry.

transformation $[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wl}$ so the right world to camera transformations is given by:

$$[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wr} = [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{c}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]_{wl} \tag{6.33}$$

In practice this is computed using the ‘product of transformations’ function given in `Tina's` math library. The code segment for this is given in Figure 6.12 for ease of reference. It should be noted that, since the initial epipolar calibration defined the two sets of external camera parameters, in terms of the initially assumed world origin which is coincident with the left camera's origin, the external transform already stored in the right camera is in fact the relative transformation.

```
..../* compute the third transformation */
    transfR = trans3_prod(*(rightcam->transf), *(leftcam->transf));
    *(rightcam->transf) = transfR;
    .........

../** from Tina's source library **/
  Transform3 trans3_prod(Transform3 transf2, Transform3 transf1)
  {
      Transform3 prod = {Transform3_id};

      prod.R = mat3_prod(transf2.R, transf1.R);
      prod.t = vec3_sum(transf2.t, mat3_vprod(transf2.R, transf1.t));
      return (prod);
  }....
```

Figure 6.12: Computing the third transformation associated with the epipolar geometry of a stereo imaging system.

With the three transformations defined, the last phase of this second stage is to optimize the internal camera parameters while minimizing the errors between the projected world calibration and corresponding image points in both the left and right cameras. This is achieved using the same optimization used with Tsai's grid calibration method except that the algorithm is applied to both the left and right cameras simultaneously. This involves adding the sum of the squared error from the optimization error functions, applied to both the left and right camera models, on every iteration. This algorithm is already included in `Tina` but has been modified for use with a single set of internal camera parameters, as described in Section 6.2.1 on page 132. This algorithm makes use of the previous error covariance from the epipolar calibration during the optimization. This is then updated by computing the full covariance matrix for all the camera parameters.

The simultaneous optimization of both the left and right cameras includes the relative camera parameters in its own parameter list. Since the accuracy of the relative camera parameters is much higher than all the other parameters at this stage the tendency in the simplex optimization algorithm is to sacrifice the accuracy of these for improvement in the rest. The final stage is therefore; to repeat the epipolar calibration, minimizing the errors in the corresponding image points and improving the estimates of the relative camera parameters. It should be noted that the restoration of the level of accuracy reported by the algorithm is not complete since the original guess values for the first epipolar calibration were calculated from the ideal and are therefore all consistent with each other.

Table 6.5 contains the results from each phase of the three stage “calibration in a box” algorithm. The left hand column of Tables 6.4(a) and 6.4(b) show the stage of the calibration algorithm for which

(a) Internal camera parameters

	Optimization errors				*Internal params*			
stage	$\sum \varepsilon^2$ *Epi*	$\lvert\bar{Epi}\rvert$	$\sum \varepsilon^2$ Rad_{dst}		u_0	v_0	a_x	f
0: *initial*					162.0	138.0	1.20	10.00
1: Epi_{min}	0.617	4.9e-07			161.9	138.0	1.17	9.997
2: f_0					“–”	“–”	“–”	0.986
2a: IP_{min}			14.51		221.7	129.9	1.07	5.471
2b: $[\mathrm{R}\,\vert\,\boldsymbol{t}]_{wr}$					“–”	“–”	“–”	“–”
2c: SIP_{min}	13.72	6.9e-04	18.17	23.84	161.9	137.8	1.18	5.453
3: Epi_{min}	7.413	-2.4e-04	19.24	22.39	162.4	138.5	1.18	5.468

(b) Relative camera parameters

	Relative camera parameters					
stage	q_1	q_2	q_3	t_1	t_2	t_3
0: *initial*	0.012	-0.035	0.035	0.0	0.034	5.233
1: Epi_{min}	0.003	0.071	0.036	2.5e-07	-0.099	5.233
2: f_0	-0.203	-0.010	0.005	0.519	0.226	356.8
2a: IP_{min}	-0.027	0.066	0.045	0.366	0.777	519.9
2b: $[\mathrm{R}\,\vert\,\boldsymbol{t}]_{wr}$	0.003	0.022	0.039	-6.4e-06	-0.088	4.734
2c: SIP_{min}	0.003	0.023	0.040	0.0	-0.078	5.251
3: Epi_{min}	0.003	0.021	0.040	6.2e-06	-0.078	4.894

Table 6.5: Results from the “calibration in a box” algorithm for (a) the Internal camera parameters and (b) the Relative camera parameters.

the results apply. This begins with initial parameters (labelled '0: *initial*'). The internal camera parameters in Table 6.4(a) are as described in previous experiments. Those in Table 6.4(b) are derived from the "known" relative rotation and translation parameters ($R_x = 0°,\ R_y = -4°,\ R_z = +4°$ and $t_x = 5.2331,\ t_y = 0.0,\ t_z = 0.18$mm from a combination with the baseline distance b in the periscopic stereo head) and mapped onto the equivalent quaternionic parameters described by Equation 6.26 earlier. The relevant optimization errors are given for each stage in Table 6.4(a) and include an estimate of the radial distortion error projected onto the image plane, as described in Section 6.3 on page144, together with the sum of the squared error for the perpendicular distance between all the corresponding image points and their respective epipolar lines (labelled '$\sum \varepsilon^2\ Epi$') and the normalized mean epipolar (labelled ' $|\bar{Epi}|$') error. The latter is included here to demonstrate the balanced, positive and negative, epipolar errors which is a good secondary indication that the calibration is likely to be accurate and stable. Visual confirmation of the calibration accuracy for the parameters of the full perspective camera models for both cameras is given in Figure 6.13. The small crosses represent the world calibration data projected onto the image frame using the calibrated camera models.

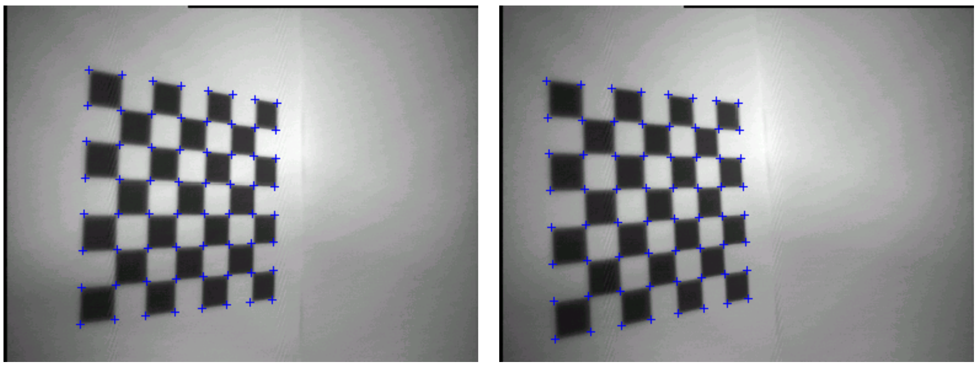

(a) left image (b) right image

Figure 6.13: Visual confirmation of calibration accuracy.

This "calibration in a box" method produces accurate, stable calibration for full perspective camera models. This can be applied to both initial laboratory calibration during manufacture and prior to use in the operating environment. Subsequent re-calibration during operation is computed using the variational principle or an alternative epipolar calibration technique.

6.4.1 Comparing the Order of the Combined Calibration Methods

In order to make a subjective comparison between the two methods of combining grid and epipolar calibration the tracking experiment described in Section 6.3 was conducted for both on the same sequence of uncorrected periscopic image data. Table 6.6 contains the results of this experiment. As

(a) 'Calibration in a box' method

order	*frames (l:r)*	*disparity (h,v)*	F_{min}	*e-band*	*matched*	$\sum \varepsilon^2 \; Epi_{min}$	$\sum \varepsilon^2 \; Rad_{dst}(l,r)$
7	20:21	20.34, -1.27	166	5	35	129.5	201.9, 346.4
5	21:22	20.30, -1.32	139	4	28	82.44	31.11, 94.69
3	22:23	19.52, -1.40	119	3	59	59.94	39.35, 24.49
1	23:24	20.38, -1.27	106	3	87	16.28	10.68, 9.410
2	24:25	20.26, -1.31	96	3	69	66.62	51.23, 46.27
4	25:26	19.81, -1.30	82	4	24	100.1	23.45, 32.33
6	26:27	20.08, -1.26	69	5	14	64.96	46.69, 12.38
8	27:28	20.25, -1.19	53	6	9	87.54	107.7, 72.51

(b) 'Grid plus epipolar' calibration

order	*frames (l:r)*	*disparity (h,v)*	F_{min}	*e-band*	*matched*	$\sum \varepsilon^2 \; Epi_{min}$	$\sum \varepsilon^2 \; Rad_{dst}(l,r)$
7	20:21	17.90, -2.64	166	5	17	40.31	16.30, 272.1
5	21:22	17.61, -2.97	139	5	24	147.6	66.58, 36.82
3	22:23	17.19, -2.77	119	3	22	49.75	23.82, 7.35
1	23:24	18.69, -2.57	106	3	86	1.963	6.92, 3.38
2	24:25	17.34, -1.95	96	3	73	43.42	30.54, 12.28
4	25:26	17.39, -2.46	82	4	39	146.4	52.84, 57.94
6	26:27	17.87, -2.54	69	5	13	63.63	16.03, 14.63
8	27:28	17.81, -2.60	53	6	3	14.72	0.29, 0.18

Table 6.6: Comparative results of calibration tracking for (a) the 'calibration in a box' against (b) the standard 'grid plus epipolar' calibration methods.

before the order the calibration sequence is given together with the relevant frames numbers. The horizontal and vertical disparity is given as a guide to the vertical shift error discussed in Section 6.3 and F_{min} is the lower of the total number of features in each images. The width of the epipolar band (labelled '*e-band*') is increased as the calibration sequence advances in order to allow the correspondence matching algorithm to recover more data as in the previous experiment. Both the sum of the squares for the epipolar and radial distortion errors are given in the right hand columns. It should be noted that the absolute values are not a good comparative measure since they are computed across all of the points, especially those of the last row in Table 6.5(b) which are from three mismatched data points. However, the balance between the radial distortion errors and the generally lower epipolar error with an increased number of matched corresponding image points at either end of the sequence demonstrates that the calibration in a box method is more stable overall than the standard method, even though the latter appears to be more accurate (has lower absolute error values for a similar number of matched points).

6.5 Concluding Remarks and Future Work

The research represented in this chapter covers both grid and epipolar calibration and a number of techniques have been reviewed.

The popular Tsai method of grid calibration was found to have a number of limitations in its use and the accuracy of its results was found to be worse than expected. Although this method has been used here, it not recommended for inclusion in a robust calibration algorithm. The alternative method reviewed in Section 6.2 is reportedly more accurate and stable, assuming the inclusion of data normalization. This method accommodates coplanar calibration data, but is not advised. The implementation of this method using a non-coplanar calibration data is therefore recommended, by the author of this dissertation.

The application of the variational principle to epipolar calibration offers a quick and simple method of maintaining calibration for small changes in the relative camera transformation. However, it's inability to estimate the internal camera parameters, up to scale, is a limitation. The current implementation makes use of all the supplied corresponding image points. Any errors, in the form of positional noise or gross outliers, in the correspondence are therefore transferred to the camera model. Although the inclusion of error covariance reduces the effect of such error over time, short term stability is a concern. Given all these facts it is recommended that the use of SVD method of computing the epipolar calibration, reviewed in Section 6.2, be investigated for the initial estimate and the variational principle used to improve the estimates and maintain the error covariance for subsequent re-calibration. However, the variational principle should be subject to the use of improved correspondence algorithm with the inclusion of the disparity gradient constraint, as in the PMF algorithm mentioned in Chapter 5, or a case deletion method such as RANSAC mentioned in Section 6.2.

The accuracy of the calibration of a stereo imaging system is improved by the use of both grid and epipolar calibration. The "calibration in a box" method presented here combines both in an apparently novel way. Simply reordering the application of grid and epipolar calibration yields accurate and stable results even with the use of less than preferred base techniques. The "calibration in a box" method involves the elementary computation of the third transformation matrix associated with the epipolar geometry of a stereo system. The author has identified that reordering the calibration and the use of elementary theory concerning the geometrical relationship between the 3D transformations do not appear to have been reported before.

Maintaining the calibration accuracy of a periscopic stereo system requires periodic epipolar re-calibration during operation. The need for re-calibration should, in theory, be low for both the rotationally corrected and uncorrected images since the relative geometry between the views is known.

However, axial offset error between the axis of rotation and the optical axis in the uncorrected data increases the need for, and reduces period between, re-calibration. In practice this error would be minimized during the manufacture. However, for the image data captured using the turn table implementation periscopic stereo head used in the course of this research such correction is not possible.

In spite of complications it is possible, using the “calibration in a box” method and subsequent epipolar re-calibration, to compute and, in theory, maintain calibration accuracy for both rotationally corrected and uncorrected periscopic data. Both can therefore be used for scene reconstruction as discussed in Chapter 7.

Chapter 7

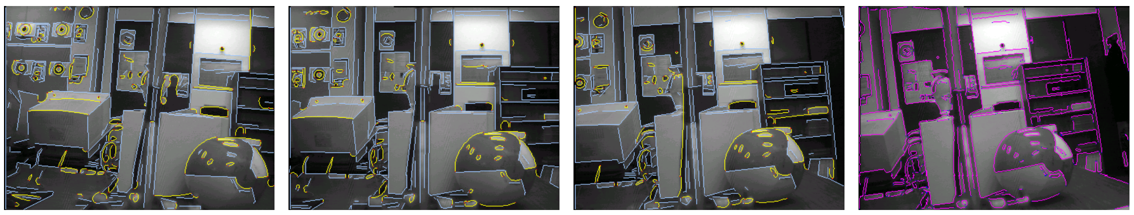
Reconstructing the Scene

In Chapters 1 and 2 it was stated that the main advantage of a periscopic stereo imaging system is that it has the capability of reconstructing large-scale 3D scenes. The long-term aim is to construct Euclidean models from which real world measurements can be derived. In order to achieve this imaged features are projected, using the camera model, to form 3D descriptions of geometric primitives such as points and lines. The type of projection and class of reconstruction depends on the type and accuracy of camera model, as described in Chapters 2 and 6. The full perspective camera and parallel camera models for the stereo system, derived in Chapter 6, enables Euclidean reconstruction of the imaged structure. However, this does not constitute a "model" of the imaged scene. A visually recognizable model of the scene requires extra processing which includes various techniques from triangulation to surface modelling. These techniques are beyond the scope of the research presented in this dissertation. Therefore, this penultimate chapter limited to a discussion of the concept of how *disparity images*, generated from periscopic stereo, are related to each other and they can be used to interactively reconstruct large-scale models of the scene.

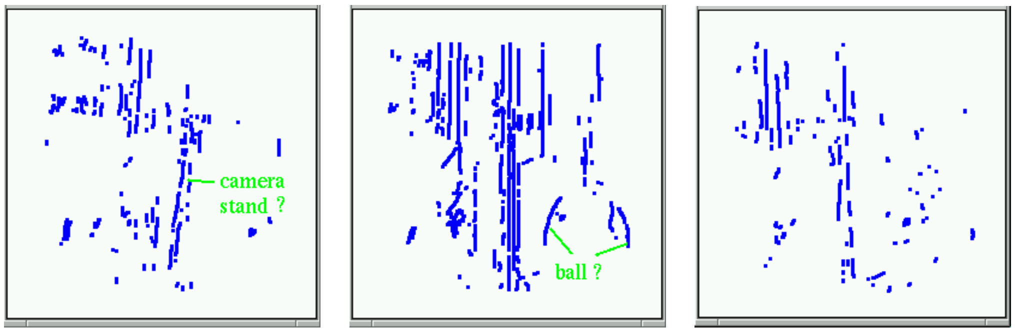
7.1 Multiple Disparity Images

In Chapter 3 the analysis of the virtual camera motion revealed that each subsequent image pairs selected from a periscopic sequence should, in theory, have the same relative geometry. In the Chapter 6, the discovery that the location of the principle point, for the camera used, was displaced from the centre of the image and the axis of rotation by a considerable margin (calibrated location estimated at pixel coordinates $(162, 138)$) represents a distinct departure from the theory for all experiments. However, the ability of the calibration tools to recover and maintain the epipolar geometry over extended periods, not withstanding the limitations of the rotationally uncorrected experimental data, supports the basic concept. The constant, deterministic, nature of the virtual relative camera geometry is compatible with the use of the temporal stereo algorithm [CLTS97], based on stretch correlation [LTM94], discussed in Section 5.4. Using this stereo algorithm, sets of disparity images can be generated for each image pair in the scan sequence as demonstrated in Figure 7.1. The disparity images in Figures 7.1(e)–7.1(g) are derived from the periscopic images in Figures 7.1(a)–7.1(d), working from left to right in pairs. The disparity is derived from the corresponding edge data by subtracting the horizontal coordinate of the right matched feature point from that of the corresponding left feature point at the intersection of the common epipolar lines [CTS98]. The disparity image is then constructed from the coordinates of the left image matches with disparity encoded in the pixel value.

On their own these disparity image convey little about the structure of the scene. It is possible to recognize certain features from the corresponding images, see Figures 7.1(b), 7.1(c) and 7.1(f). However, this is more connected with the ability of the human mind to resolve structure by interpolation rather than the accuracy or content of the projection. The disparity images consist largely of vertical structure. This is due to the predilection for the use of non-horizontal edges in the temporal stereo algorithm. Horizontal edges can not be used to derive a measure of the disparity since the error is



(a) frame 20 (b) frame 21 (c) frame 22 (d) frame 23

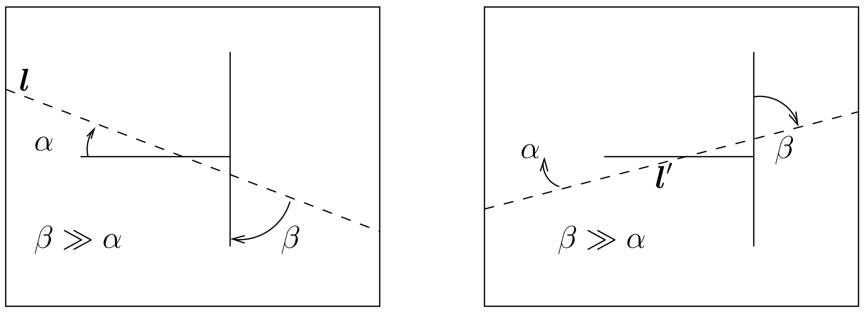


(e) disparity image f.20:21 (f) disparity image f.21:22 (g) disparity image f.22:23

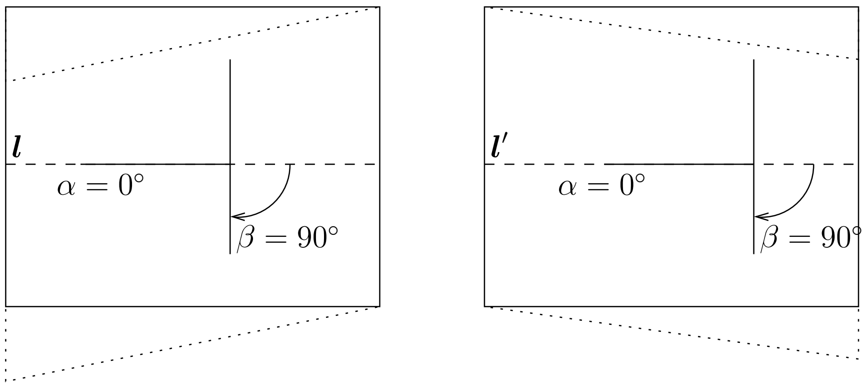
Figure 7.1: Sequential disparity images (e)–(f) from uncorrected periscopic image data (a)–(d).

inversely proportional to the angle between the edge and the common epipolar line created by the image plane rectification for the stretch correlation technique [CTS98], as shown in Figure 7.2. It should be noted that the diagrams in Figure 7.2(b) are not entirely accurate since the structure should be "warped" in sympathy with the image plane and a more accurate definition of the problem is that scene in the scene which is parallel to corresponding epipolar lines can not be used for disparity measurement. This is a distinct limitation to the reconstruction of large-scale models since a considerable percentage of the structure in the human engineered world is horizontal as well as vertical. However, this limitation is addressed in Section 7.2.

Each disparity image is projected from the left camera, which is taken to be the reference coordinate frame. The camera frame, either left or right, is the only frame of reference possible at this stage. However, using knowledge of the sequence number of the original images, the disparity images



(a) Non-rectified image planes



(b) Rectified image planes

Figure 7.2: Lost of the disparity measure for horizontal structure.

can be projected on to an absolute coordinate frame centered on the periscope system origin. This is defined as the intersection of the optical axis and mirror plane in Section 3.1. The origin for the reconstructing coordinate frame is therefore given by the preset distance b between optical centre and the mirror plane as defined in Section 3.1, see Figures 3.3 and 3.7. The transformation to this coordinate frame is effectively a rotation followed by translation into the scene along the optical axis such that each projected 3D point is therefore given by:

$$\boldsymbol{X}' = \mathrm{R}\,\boldsymbol{X} + \boldsymbol{t} \qquad \text{where} \qquad \mathrm{R} = \begin{bmatrix} \cos m\phi & 0 & \sin m\phi \\ 0 & 1 & 0 \\ -\sin m\phi & 0 & \cos m\phi \end{bmatrix} \tag{7.1}$$

$\boldsymbol{t} = [0,\ 0,\ b]$, m is the frame number and $\phi = 4°$, from the mirror rotating at 16.6 rpm with a capture rate of 25 fps, as described in Section 3.4. R is a 3D rotation about the *Y-axis* of a right handed coordinate frame. This effectively forms a panorama of over lapping disparity images as demonstrated in Figure 7.3

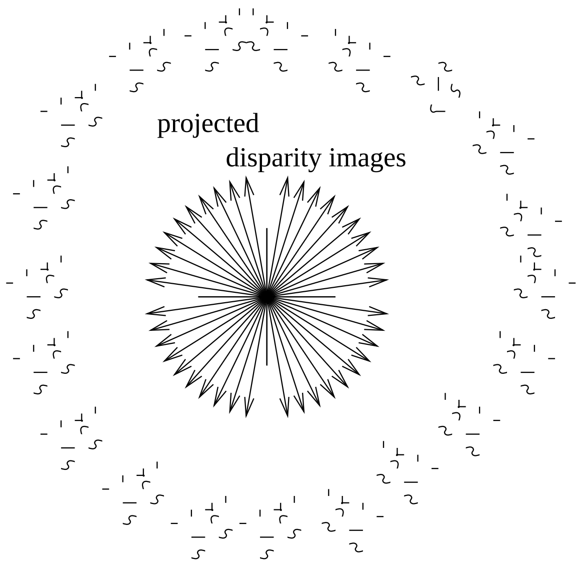


Figure 7.3: Panorama of projected disparity images.

With approximately 4° separating each image there are 90 images in each horizontal scan of the surrounding environment. Using successive images yields 90 disparity images for each scan. Each successive image pair produces a disparity image which is consistent with every other, since they are projected using the same camera models. This assumes "reasonably" stable internal and relative camera calibration but not absolute accuracy or stability. This statement is explained Section 7.3 later. Each projected cloud of 3D data is therefore related by a Euclidean transformation. Knowledge of this transformation could be used in some form of data fusion algorithm which combines the over lapping sets of projected 3D data.

The creation of this panorama of projected data assumes that each of the disparity images were derived from fronto-parallel images. The mirror rotation about the optical axis in uncorrected periscopic data will however be transferred to the disparity image and the projected data, as evident in Figure 7.1(f). The rotational correction of the periscopic data must therefore be taken into consideration.

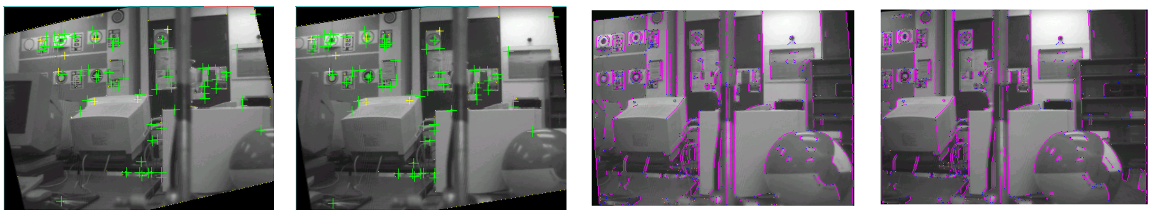
7.2 Late Correction of the Periscopic Rotation

The concept of rotationally corrected and uncorrected periscopic data has continued throughout this dissertation. The reason for this is due to three aspects, two of which have already been discussed in Sections 3.2 and 5.3. To summarize; the first was that interpolation of image data for correction induces positional error in the feature localization. The second concerns the complexity of processing through the rotating silhouette frame. This is evident in Figure 7.4 where the silhouette frame appears in the projected data in the disparity images shown in Figures 7.4(e) and 7.4(f). Figures 7.4(a) and 7.4(b) show images from corrected periscopic images, processed for re-calibration. The images in Figures 7.4(c) and 7.4(d) are produced by the stretch correlation algorithm and are therefore distorted[1] (the ball appears oval) from the application of the image plane rectification, as discussed in Section 5.4.

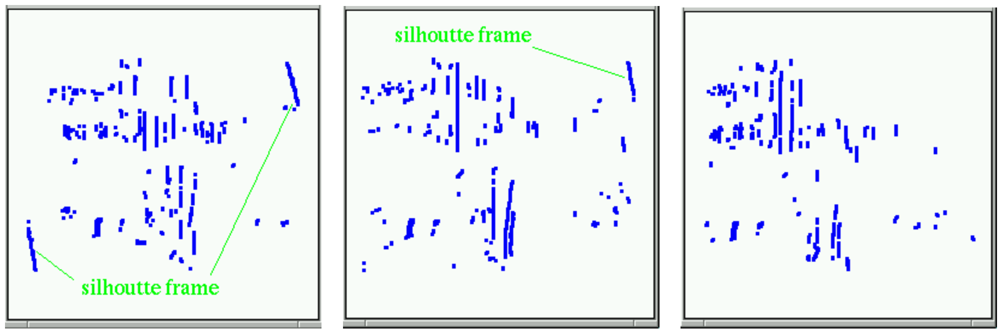
Processing the rotationally uncorrected periscopic data has presented its own problems, as discussed in Chapters 5 and 6. However, using this data has an extra hidden bonus. Since the image data is constantly rotating all horizontal structures will at some instance in time be imaged as non-horizontal lines. The use of uncorrected periscopic data can therefore compensate for the limitation in the temporal stereo algorithm previously discussed in Section 7.1.

The rotational correction can be combined with rotation matrix in Equation 7.1 and applied to the projected data in 3D. Alternatively rotational correction can be performed in 2D as demonstrated in Figure 7.5. This rotation shown is deliberately excessive in order the convey the concept. This was

[1]The distortion caused by the image plane rectification has been included here for reference only and has no special significance to the discussion in this chapter.

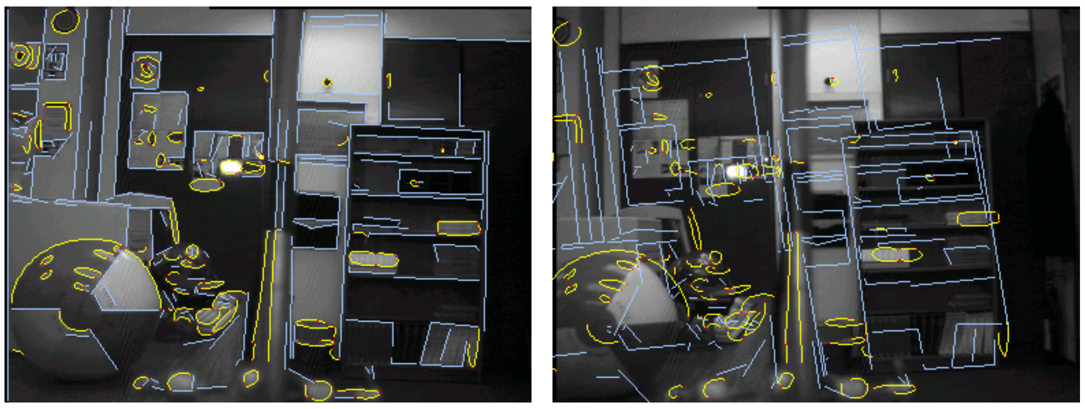


(a) cal of f.18 (b) cal of f.19 (c) stretch_corl f.20 (d) stretch_corl f.21



(e) disparity image f.18:19 (f) disparity image f.19:20 (g) disparity image f.20:21

Figure 7.4: Disparity images from rotationally corrected periscopic data with: (a) and (b) the calibration corners, (c) and (d) a temporal stereo pair after stretch correlation and matching and (e)–(f) sequential disparity images from the image pairs (a):(b),(b):(c) and (c):(d).



(a) matched features overlaid on left image (b) rotated geometry list from matched features

Figure 7.5: Rotating the list 2D geometry, create from the matched features, before projection to 3D.

created by applying a 2D rotation matrix, of the form:

$$R_{\phi z} = \begin{bmatrix} \cos m\phi & \sin m\phi \\ -\sin m\phi & \cos m\phi \end{bmatrix} \tag{7.2}$$

to the geometry shown in Figure 7.5(a). It is not suggested that re-projection into the right hand image is necessary or correct; this only demonstrates the concept of correction in 2D[2].

Although the panorama has been projected into an absolute coordinate frame the accuracy of the reconstruction, regardless of the data fusion and modelling techniques employed, will still have poor accuracy. This is due to the small baseline inherent in the design of the periscopic stereo head. It is well known the accuracy of depth estimation is inversely proportional to the baseline distance between two views [HZ00]. Even if every second or third frame is used to form an image pair and the disparity images produced were combined with the original panoramic data, the accuracy would still be poor compared with a conventional stereo reconstruction. However, this is addressed in the next section.

7.3 Multiple Periscopic Scans

The single panorama of projected data described in Sections 7.1 and 7.2 does not constitute large-scale reconstruction. It does not yield a navigable map of the local environment, nor does it allow for accurate measurements of world structures.

In Chapter 1 it was stated that the target application for periscopic stereo is for autonomous mobile robots. Therefore, the periscopic head can be relocated in a new position displaced by some distance and a second scan of the local environment captured. The sketch in Figure 7.6 demonstrates two successive scans of a local environment. The relative displacement, D, could be recovered from

[2]A list of 2D geometry is constructed in `Tina` using conic and line fitting techniques applied to the edge strings in the left hand image

odometers on the robot. However, this is not essential since the relative camera positions can be recovered by epipolar calibration between corresponding image data across the two sequences. Computing

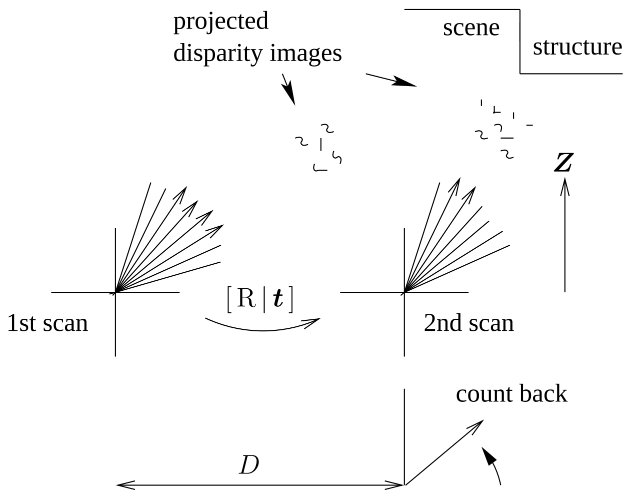


Figure 7.6: Registration of projected disparity images from different scans.

the epipolar calibration across different scan image sequences allows for the reconstruction of a third set of projected data. This, together with the data from each scan, can be combined to form more accurate reconstruction of both depth and structure. Subsequent scans of the local environment would recover yet more data and a large-scale model of the scene is produced incrementally over time. For each subsequent scan the epipolar calibration is also used to transform the data from the first absolute coordinate frame to the second, as shown in Figure 7.6. This effectively creates a transitional world origin which is updated by every new position. Using the last position of the periscopic stereo head as the origin of the coordinate frame, navigation back through the world model and absolute scale measurement are both possible.

Registration of the corresponding images can be achieved by initially using a crude “count back” method of the frames and then subsequently by using the feature matching techniques described in Chapter 5. This is possible since each scan is captured from the same cardinal point, as described in

Section 3.4. The number of frames to count back in the second scan, relative to the first, as shown in Figure 7.6, is proportional to the baseline distance between the scans and the mean depth of the scene. Both of these can be recovered from the data computed thus far. A secondary support for the registration can be derived from the percentage of corresponding image points similar to the measure used in Section 5.3.

Figure 7.7 demonstrates the generation of a disparity image from widely disparate views. Again,

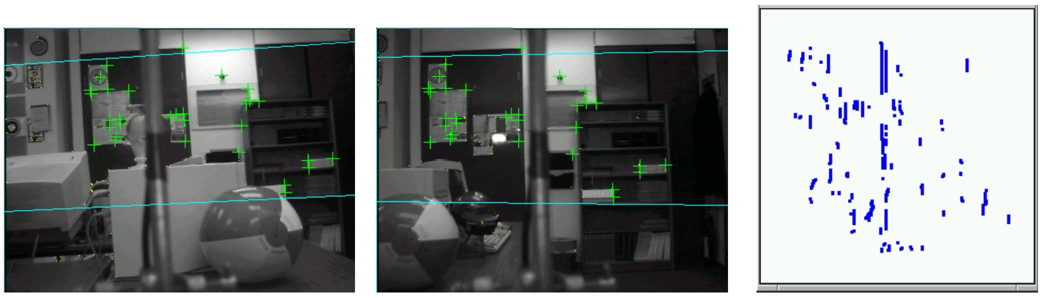

(a) frame.22, seq.2 (b) frame.24, seq.1 (c) disp_image

Figure 7.7: Disparity image from registered images across different periscopic sequences.

this makes use of the temporal stereo algorithm [CTS98]. The original images in Figures 7.7(a) and 7.7(b) show the corresponding points used for re-calibration and two arbitrary epipolar lines are given to convey the calibration accuracy. Using the variational principle for epipolar calibration with the error covariance was found to give accurate and stable results after the initial phase. However, the initial calibration for a unknown transition of the periscopic head is largely dependent of the validity of the correspondence which is inversely proportional to the distance moved by the head.

7.4 Closing Remarks

This chapter has presented the outline of an iterative method of reconstructing large-scale models of the surrounding scene. It is based on the projection for disparity images onto a common world coordinate frame which is updated by the transition of the periscopic head to subsequent positions in the local environment.

The proposed method makes use of the temporal stereo algorithm since it is compatible with the consistent relative, virtual, camera geometry inherent in periscopic stereo. The use of the stretch correlation technique has distinct advantages for widely disparate views from subsequent scans. However, from the discussion in Section 5.2, the use of the stretch correlation technique for the individual scans may not be necessary or desirable. Although the use of successive images is described, there is sufficient overlapping in the image data of every second frame. This allows for the possibility of the inclusion of disparity images from a wider baseline and therefore improved depth estimation and the possibility of processing across three views using recently developed techniques involving the use of tensors. This reportedly [HZ00, chap.16–18] increases both depth accuracy and structural content.

The final advantage of the late correction of the rotation about the optical axis in periscopic data directly addresses the limitation in the temporal stereo algorithm for reconstruction of horizontal structure, or more accurately, structure in the scene which is parallel to the corresponding epipolar lines. With subsequent scans from disparate viewpoints, accurate reconstruction of horizontal structure would increase over time. This would naturally lag behind accuracy and volume of non-horizontal structure.

Much of the discussion presented in this chapter is conjecture on the part of the author since proof of the construction of large-scale 3D models can only really be given by their existence. However, this conjecture is based on “known” theory from computer vision and robotic navigation using triangulation. The initial experiments have shown the methodology to be correct. However, the large errors induced by the offset of the centre of rotation from the optical axis for the system used make the reconstruction of large-scale models difficult with the current data. Final proof of the ability of periscopic stereo to reconstruct large-scale models of the imaged scene is therefore the of subject of ongoing research.

Chapter 8

Conclusions and Future Work

The research presented in this dissertation has taken the concept of periscopic stereo through to the design and construction of a realizable imaging system. The capability of periscopic stereo for large-scale 3D reconstruction of the imaged scene has been demonstrated via the examination of each stage of the reconstruction process.

In Chapter 3 the analysis of the virtual camera geometry revealed that, assuming rotation about the optical axis, relative motion between the views is consistent. The virtual baseline distance depends on the size of the head and the choice of camera lens. Synchronization of the speed of rotation, the camera shutter speed and the image capture rate is recommended since it ensures constant relative motion of the camera. The induced “tumbling” motion inherent in periscopic stereo image data presents two distinct approaches for processing the image data. One consists of initially applying a 2D rotation to the image plane in order to remove the tumbling effect and then processing the data using standard stereo algorithms. However, this “corrected” approach induces a rotating “silhouette frame” into the image sequence which creates it’s own problems for bounding the region of interest during correspondence matching, calibration and reconstruction processes. The second, “uncorrected”, approach makes no attempt to initially correct for the tumbling effect and processes the image data using suitably modified stereo algorithms in order to implement rotational invariance or compensate for it. In most cases, the modifications are relatively minor since the amount of relative rotation between the frames is small. In either case, the tumbling effect has been central to the consideration of each stage of the overall scene reconstruction process.

The positional accuracy of the reconstructed scene points is dependent on the accuracy of the camera projection matrix and the localization of the corresponding image points. Feature detection to sub-pixel accuracy is necessary for positional accuracy since localization error leads to the fracture of edge strings and to the possibility of mismatched correspondence data. In Chapter 4 the SUSAN algorithm for feature detection was found to be an efficient and effective alternative to derivative based methods. The implementation of SUSAN edge and corner detection reveals complications. Identifying possible edge errors near junctions lead to the concept of "tuning" the output response for connected, or segmented, edge strings. Modifications to the implementation of the SUSAN edge detector have been evaluated. The results show that the optimal threshold for the secondary "edge-type" condition, defined by the magnitude of the centre of gravity vector of 0.65, improves efficiency of the algorithm without incurring any degradation of performance. The inclusion of an additional parameter, valid over a limited range, effectively allows for the binary selection for edge connectivity. However, the integration of SUSAN feature detection with subsequent processing is not seamless since the definitions of feature contrast and orientation are different from derivative based equivalents. A proper statistical analysis is required to fully validate the modifications made and assess the accuracy of sub-pixel localization.

Matching corresponding features in two or more images is an ambiguous problem with the preferred solution dependent on the particular set of circumstances regarding the imaging system, the scene and subsequent processing requiring the match. An optimal solution for all cases does not exist since the application of particular constraints is not the same in all cases. Given the requirement for matched point features from periscopic data for subsequent re-calibration, the preferred solution employs the correspondence algorithm available in `Tina` with the choice of two correlation techniques. In the case of uncorrected periscopic data a $\pm 4^\circ$ rotated correlation patch is marginally preferable to a standard correlation patch. In the case of corrected periscopic data the warped patch, guided by feature orientation found in `Tina's` library of source code, is recommended since it is less affected by the induced silhouette frame.

These "patch-warping" correlation techniques are preferred to the proven stretch-correlation technique since no actual rectification of the image plane is required. Preference, in this particular case, is based on the use of simulated rectification of the image plane which does not distort the localization of the point features. Therefore, the accuracy of the epipolar calibration is partially decoupled from the collection of accurate data for which it is required. However, in the case of the matching correspondence data for reconstruction, the use of stretch-correlation in the temporal stereo algorithm is preferred since the reduction in mismatched data, from the use of temporal disparity gradient information, outweighs the degree of localization error induced. The assumption in this argument is that occurrence of mismatched data in the correspondence data set does not unduly affect the calibration accuracy. This is a questionable premise since the reported stability of the variational principle for epipolar calibration in the presence of outliers is not been supported by the results presented in Section 6.3.

Currently two schools of thought exist on the question of outliers and calibration. One suggests that all outliers should, as far as possible, be removed from the correspondence data prior to calibration regardless of the optimization strategy used. The other claims that correctly constrained optimization negates the need for computationally intensive case deletion methods. The author of this dissertation prefers a compromise approach where the correspondence algorithm is sufficiently constrained to reduce the number of outliers without the need for further case deletion methods and the calibration algorithms are sufficiently robust in the presence of a small number of outliers. The use of the term “sufficiently” here is deliberately vague since an accurate measure in either is impossible due to the inability to predict the number, or degree of error, of outliers in the general case. The author’s preference is merely an observation and unsubstantiated here.

The inadvertent inclusion of outliers in the correspondence data from uncorrected periscopic data highlights the need for a disparity limit constraint, based on the known relative camera parameters, and/or a disparity gradient limit. However, the latter is known to be unstable for point correspondence data since uniform distribution of the data across the image is unlikely. Modelling the axial offset error induced in the periscopic data by the limitations of the simulated system has produced a recommended estimate of the maximum allowable alignment error of ± 5 pixels. This is within current manufacturing capability.

A number of calibration techniques were reviewed in Chapter 6. Although Tsai’s method of grid calibration is widely referenced, the results from experiments with a coplanar grid found it to be unstable and inaccurate, especially when the initial guess for the principle point is in error by more than a few pixels. Assuming a reasonably accurate estimate of the principle point, Tsai’s method is more stable with an iterative approach to the optimization phase. The preferred sequence begins with optimization for the focal length only, followed by the inclusion of the principle point with the horizontal aspect ratio. However, the accuracy of the results is still questionable and in general Tsai’s method of grid calibration is unsuitable for an autonomous calibration system.

Epipolar calibration using the variational principle was shown to yield accurate stable estimates of the relative camera parameters. However, its ability to derive estimates of the internal parameters from an initially uncalibrated state is poor since it is constrained to achieve an optimal result by gradual iteration. Stability in the presence of outliers can not be concluded at this time. In the case of uncorrected periscopic data, the apparent performance is good considering the prevailing experimental conditions. However, calibration accuracy can not be maintained for more than six successive image pairs. In the case of widely disparate views, when registering images from different scans, the ability to acquire and maintain calibration is more apparent with the variational principle. However, this has not been formerly confirmed. The inclusion of an initial estimate of calibration using the linear, closed-form, solution given by SVD may lead to more stability in the both epipolar and grid calibration.

A novel combination of epipolar and grid calibration for initial calibration, prior to use, referred to as "calibration in a box" has been presented. The use of a self contained system does not invalidate the concept of remote operation and the concept of scaling the grid calibration data for the intended scene depth is possible. Calibration in a box switches the traditional order in which grid and epipolar techniques are applied. Instead, a three stage process that begins and ends with epipolar calibration, via optimization using the variational principle, is used. This algorithm relies on an elementary geometric combination of the three transformation matrices inherent in a stereo camera system, where the third is derived from knowledge of the other two. Although not extensively tested, this algorithm yields stable calibration results at the first attempt since it reduces the error in the initial camera parameters for the focal length and aspect ratio. However, the accuracy is largely dependent on "base" methods for the grid and epipolar calibration either now which were found to give reliable estimates of the image plane coordinates for the principal point. Although the external parameters of the right hand camera are calculated directly, a final optimization stage is essential to return the accuracy of the relative camera transformation and ensure stability for subsequent epipolar re-calibration during operational use. Improved results are achieved by allowing multiple scans of the periscopic head inside the box with successive optimization for the relative and external parameters. Each of which should be accompanied by an appropriate error covariance estimate. Re-calibration of the system is conducted in the normal manner using corresponding image data and epipolar calibration. Control of periodic re-calibration could be given either by a correspondence performance measure and/or some random event within a maximum period (say once every scan on a randomly selected pair of images).

The theory of large-scale scene reconstruction using disparity images from periscopic data has been presented. Using knowledge of the frame number and the calibrated camera projection matrix allows for the transformation of the disparity image data onto the world coordinate frame defined by the system origin. Successive disparity images from a periscopic scan can be projected to form a panorama. On their own panoramas of projected disparities images are not particularly accurate. However, the generation of disparity images from periscopic images registered across separate scans yields more accurate depth estimation. The combination of separate panoramas and "cross-scan" disparity images should enable accurate reconstruction of the surrounding scene in an efficient manner that directly addresses the problem of "where to look next". The late correction of the tumbling motion, inherent in periscopic data, has a number of advantages. It removes the need for initial interpolation of the image data together with the induced feature localization error. It simplifies image processing by circumventing the need for processing through the silhouette frame. The apparent complexity of processing tumbling image data was shown to be unfounded and in the case of the use of the temporal stereo algorithm directly addresses the limitation of reconstructing structure in the scene which is parallel with corresponding epipolar lines. In other stereo systems this limitation is given as structure which is horizontal in the scene. Apart from improvements in the calibration and correspondence algorithms, future work on large-scale reconstruction will undoubtedly concentrate on the fusion of overlapping structure in the panorama from adjacent projected disparity images, cross panoramic depth and structural information.

Most of the techniques employed in the course of this research are based on, existing stereo vision techniques. However, virtually all have required some extension or modification in the context of periscopic stereo. Individually these techniques offer little which could be regarded as "new" theory, with the possible exception being the “calibration in a box” concept. However, in combination the techniques presented make periscopic stereo practical for the extrication of 3D structure and an effective compromise compared to existing imaging systems, by simplifying many of the problems inherent in scene reconstruction. Periscopic stereo is the only system proposed to date which could be capable of producing autonomous large-scale scene reconstruction, efficiently. However, the lack of a complete demonstration of registered panoramas of disparity images means that this conclusion can not be fully substantiated at this time. It is suggested that the addition of periscopic stereo head to a robot’s sensor array may, for example, not only allow for increased robotic perception by supplying an “awareness” of the surrounding environment but subsequently allow telepresence exploration of remote mapped environments.

Appendix A

Perspective Projection, Coordinate Systems and Epipolar Geometry

This tutorial briefly describes the basic theory of the perspective projection performed by a camera in terms of the transformations across the relevant coordinate systems. It also covers the geometry between two cameras which form a stereo imaging system. Alternative tutorials can be found via CVOnline [1].

The imaging system of a camera performs a perspective projection (also called central projection) on the world such that all points along a line from the optical centre out into the scene are projected to a single image point. The effect is more apparent when viewing two parallel lines in the 3D world which appear to converge to a point at infinity, known as the *vanishing point*. This projection is essentially a linear transformation of a 3D projective space ($\mathcal{P}^3$), the world, to a 2D projective space ($\mathcal{P}^2$), the image plane. However, if the points in these spaces are represented with normal, Euclidean coordinates the expressions, that are used to define the transformations, become non-linear.

[1] The CVonline website can be found at:- http://www.dai.ed.ac.uk/CVonline/

In order to keep the geometry simple, yet allow for the mathematical definition of primitives such as points and lines at infinity, projective coordinates, often called *homogeneous* coordinates, are used for the analysis of projective transformations. The vector defining a normal point represented in homogeneous coordinates has an extra component, which can take any non-zero value. This is often set to '1' and a one-to-one mapping from Euclidean space ($\mathcal{R}^n$) into projective space ($\mathcal{P}^n$) is therefore given by:

$$[x_1\,, \ldots\,, x_n]^T \longrightarrow [x_1\,, \ldots\,, x_n\,, 1]^T$$

Points at infinity, often referred to as *ideal points*, have no Euclidean representation, but are defined in homogeneous coordinates by $[x_1\,, \ldots\,, x_n\,, 0]^T$. Two homogeneous points are equivalent if one is a scalar multiple of the other, $(x_1\,, x_2\,, 1)^T \equiv (\lambda x_1\,, \lambda x_2\,, \lambda)^T$. A *collineation*, or projective transformation, is any mapping, $\mathcal{P}^n \rightarrow \mathcal{P}^n$, which can be defined by an $(n+1) \times (n+1)$ matrix A, such that $\tilde{\boldsymbol{p}} = A\,\tilde{\boldsymbol{q}}$ where $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{q}}$ are the mathematical shorthand for homogeneous coordinates given by $(n+1)$ vectors.

Figure A.1 demonstrates the geometry for the perspective projection performed by a camera. Three coordinate systems are used to model the projection of a point in the world, defined by a 3-vector ($\boldsymbol{X}_w$) in the world coordinates, to a point on the image plane, defined by either a homogeneous 3-vector $\boldsymbol{X}_i = [X_i\,, Y_i\,, Z_i]^T$, or its 2D equivalent given by

$$\boldsymbol{x}_i = [u, v]^T = [X_i/\,Z_i\,, Y_i/\,Z_i]^T \tag{A.1}$$

The three coordinate systems are:

1. The *World coordinate system* (subscript $_w$) describes the position of objects in the world with respect to a defined origin $\mathbf{0}_w$.

2. The *Camera coordinate system* (subscript $_c$) describes the position of objects with respect to its own optical centre $\mathbf{0}_c$, sometimes defined as the focal point **C**.

3. The *Image coordinate system* (subscript $_i$), which has its axis aligned to the camera coordinate system, describes the position of the imaged point in terms of the pixel coordinates in the image plane.

It should be noted that an extra *image affine coordinate system* is sometimes used to distinguish between the ideal coordinates and the actual coordinates when imperfections in the geometry of the pixel grid are taken into account.

The scene point, in the world coordinate system, can be translated into the camera coordinate

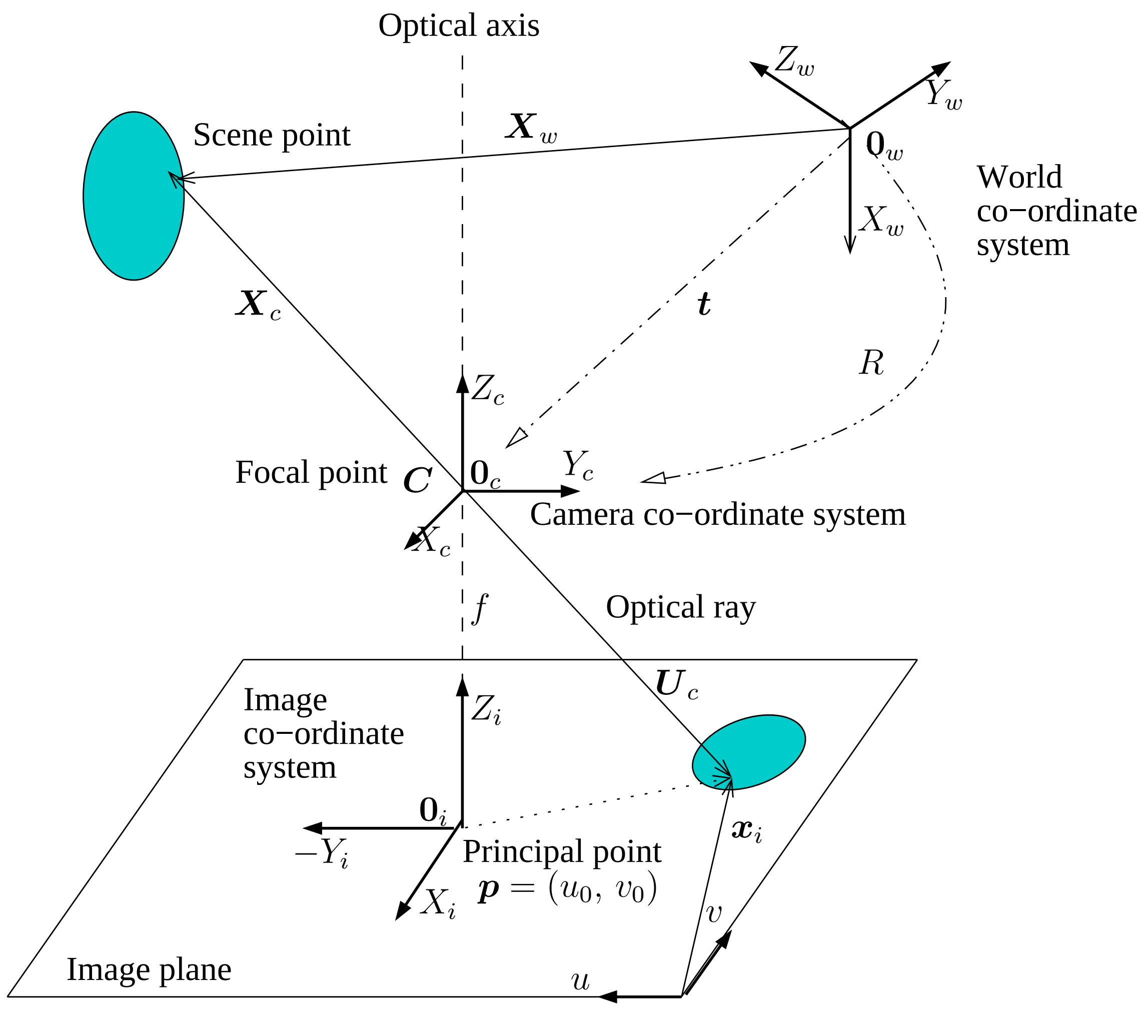


Figure A.1: Basic geometry for perspective projection.

system by a translation vector ($\boldsymbol{t}$) followed by a rotation matrix (R) such that $\boldsymbol{X}_c = \mathrm{R}(\boldsymbol{X}_w - \boldsymbol{t})$ as shown in Figure A.1. These are often known as the *extrinsic*, or external, camera parameters and can be given in the form,

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = [\,\mathrm{R}\,|\,\boldsymbol{t}\,] \tag{A.2}$$

where the delimiter [|] is used to denote that the matrix is composed of two sub-matrices, or, in this case, a matrix and a vector.

The world point $\boldsymbol{X}_c$, given in the camera coordinate system, is projected on to the image plane as point $\boldsymbol{U}_c$ where the x_c and y_c coordinates can be derived from similar triangles, as shown in Figure A.2, such that:

$$\boldsymbol{U}_c = \left[\frac{f\,X_c}{Z_c}\,,\ \frac{f\,Y_c}{Z_c}\,,\ f \right]^T \tag{A.3}$$

where f is the *focal length* between the optical centre of the lens and the image plane. This projection can be represented by a linear mapping between homogeneous coordinates using the normalized *projection matrix* M as:

$$\boldsymbol{U}_C = \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \lambda \mathrm{M}\,\boldsymbol{X}_c \tag{A.4}$$

where λ ($\lambda = f/Z_c$) is the scale factor.

The focal point $\boldsymbol{C}$ is the centre of the camera projection and the origin of the camera coordinate system $\mathbf{0}_c$. In Figures A.1 and A.2 the focal point appears in front of the image plane, which is its actual physical position. However the focal point is often shown behind the image plane in order to simplify the explanation of projection. In that case the vector $\boldsymbol{U}_c$ differs only by the sign of f.

The final translation is from the camera coordinate system to the image coordinate system as shown in Figure A.3. The *principal point*, at coordinates $(u_o\,,\ v_o)$, defines the point where the optical axis intersects the image plane. The origin for the pixel coordinates is however defined as the top left-hand corner of the image plane. The translation is therefore given by,

$$k_u\, x_c = u - u_o \qquad \text{and} \qquad k_v\, y_c = v_o - v \tag{A.5}$$

where the units of k are in pixels/length as shown in Figure A.3.

Combining with f from Equation A.4, Equation A.5 can be expressed in matrix form by,

$$\boldsymbol{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\,k_u & 0 & u_o \\ 0 & -f\,k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \mathrm{K} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} \tag{A.6}$$

where K is known as the *camera calibration matrix* and is a 3×3 upper triangular matrix. This is

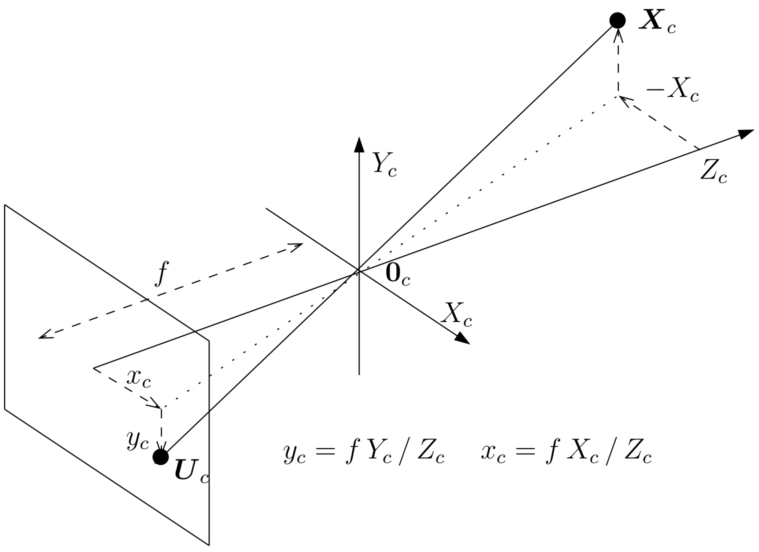


Figure A.2: Derivation of image point from similar triangles.

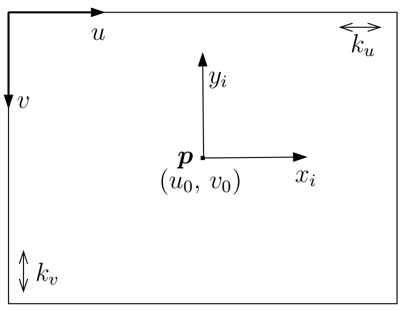


Figure A.3: Conversion to image coordinate system.

more usually defined as:

$$\mathrm{K} = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \tag{A.7}$$

where $\alpha_u = f\, k_u$ and $\alpha_v = -f\, k_v$ define the scaling in the image x and y directions and as such the *aspect ratio*, α_u/α_v, of the pixel elements. If the pixel grid is assumed to be perpendicular, the above is satisfactory however sometimes an extra term (α_{sh}) is shown in the top row, centre column which describes a *shear* factor. The components of the camera calibration matrix are collectively known as the the *intrinsic*, or internal, camera parameters.

Combining all three translations above yields the 3×4 projection matrix P which models the transformation from 3D Euclidean space to an image.

$$\boldsymbol{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{R} & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \mathrm{KM}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]\ \boldsymbol{X}_w \tag{A.8}$$

This is often simplified to $\tilde{\boldsymbol{x}}_i = \mathrm{P}\tilde{\boldsymbol{X}}_w$ where $\mathrm{P} = \mathrm{KM}\ [\,\mathrm{R}\,|\,\boldsymbol{t}\,]$ as shown. Often only the form of (P) is important and not its actual decomposition. However, decomposition can be achieved using a matrix factorization method such a **QR** decomposition which can be found in *Numerical Recipies* [PFTV93].

Two-View Geometry

Using two cameras to form a stereo imaging system, the 3D coordinates of any world point, viewed in both cameras, can be computed from the intersection of the image rays projected from the optical centers of each camera. The geometry that relates the two views of a stereo camera system is known as *epipolar* geometry and is shown in Figure A.4.

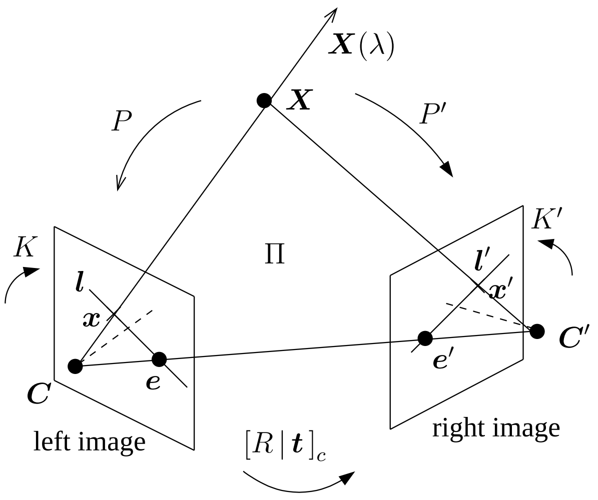


Figure A.4: Epipolar geometry for a stereo camera system.

The abstract, or conceptual, plane (Π) is defined by the optical centers of both the left and right camera and the world point of interest and is known as the *epipolar plane*. This plane, intersects each image plane creating an abstract line, called the *epipolar line*. Each epipolar line, in turn, defines two points; an image point coincident with the projected image ray and an *epipole*. The epipoles ($\boldsymbol{e}$ and $\boldsymbol{e}'$), are the projection of the optical centers, viewed by the other camera, in both the right and left images respectively. These points are not necessarily on the image plane, as is the case with parallel cameras. Two cameras with coplanar image planes and parallel optical axis are referred to as the *canonical configuration* and produce collinear epipolar lines.

Any two image points corresponding to the same 3D world point are constrained to lie on the epipolar line in the other image. This forms the basis of solutions to stereo matching, calibration and subsequently 3D scene reconstruction. The epipolar geometry of a stereo camera system is uniquely described by the *Fundamental* matrix such that:

$$\tilde{\boldsymbol{x}}^T \mathrm{F}\, \tilde{\boldsymbol{x}}' = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} & & \\ & \mathrm{F} & \\ & & \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{0} \tag{A.9}$$

where $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}'$ are corresponding, homogeneous image points and F is a 3×3 matrix.

Proof that the epipolar geometry of a stereo system is described by the fundamental matrix is derived as follows. Consider the image ray $\boldsymbol{X}(\lambda)$, back-projected from $\boldsymbol{x}$, on the left image plane in Figure A.4, by the projection matrix P. This is obtained from $\tilde{\boldsymbol{x}} = \mathrm{P}\tilde{\boldsymbol{X}}$ by considering two points of interest on the image ray; the optical centre of the camera $\boldsymbol{C}$, where $\mathrm{P}\boldsymbol{C} = \mathbf{0}$, $\lambda = \infty$, and the point $\mathrm{P}^+\boldsymbol{x}$, where $\lambda = 0$ and P^+ is the pseudo inverse of P such that $\mathrm{P}^+ = \mathrm{P}^T(\mathrm{P}\mathrm{P}^T)^{-1}$ for which $\mathrm{P}\mathrm{P}^+ = \mathrm{I}$. Assume that the world origin is coincident with the left camera centre such that $\mathrm{P} = \mathrm{K}\,[\,\mathrm{I}\,|\,\mathbf{0}\,]$ and $\mathrm{P}' = \mathrm{K}'\,[\,\mathrm{R}\,|\,\boldsymbol{t}\,]$ The image ray is then line formed by the two points and given by:

$$\boldsymbol{X}(\lambda) = \mathrm{P}^+\boldsymbol{x} + \lambda\boldsymbol{C} = \begin{bmatrix} \mathrm{K}^{-1}\,\boldsymbol{x} \\ \mathbf{0}^T \end{bmatrix} + \lambda \begin{bmatrix} \boldsymbol{C} \\ 1 \end{bmatrix} \tag{A.10}$$

Consider these two points imaged by the second camera such that:

$$\boldsymbol{x}'_{\lambda=\infty} \;=\; \mathrm{P}'\boldsymbol{C} \;\simeq\; \mathrm{K}'\,[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_c \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \;=\; \mathrm{K}'\,\boldsymbol{t} \;=\; \boldsymbol{e}' \tag{A.11}$$

$$\boldsymbol{x}'_{\lambda=0} \;=\; \mathrm{P}'\mathrm{P}^+\boldsymbol{x} \;\simeq\; \mathrm{K}'\,[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_c \begin{bmatrix} \mathrm{K}^{-1}\,\boldsymbol{x} \\ 0 \end{bmatrix} \;=\; \mathrm{K}'\,\mathrm{R}\,\mathrm{K}^{-1}\,\boldsymbol{x} \tag{A.12}$$

where $\simeq$ denotes the equivalence *up-to-scale*, $\boldsymbol{e}'$ is the epipole and $[\,\mathrm{R}\,|\,\boldsymbol{t}\,]_c$ is the transformation from the world origin at the left camera centre. The ray is imaged in the second camera as the epipolar line and can be computed from the vector product

$$\begin{aligned} \boldsymbol{l}' &= \boldsymbol{e}' \times \boldsymbol{x}' && \text{(A.13)} \\ &= \left[\,\mathrm{P}'\boldsymbol{C}\,\right]_\times \mathrm{P}'\mathrm{P}^+\,\boldsymbol{x} \\ &= \mathrm{K}'\boldsymbol{t} \times \mathrm{K}'\mathrm{R}\,\mathrm{K}^{-1}\,\boldsymbol{x} \\ &= \mathrm{K}'^{-T}\,[\boldsymbol{t}]_\times\;\mathrm{R}\,\mathrm{K}^{-1}\boldsymbol{x} && \text{(A.14)} \end{aligned}$$

where $[\;]_\times$ denotes the skew-symmetric matrix form of the translation vector given by:

$$[\boldsymbol{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{A.15}$$

A similar algebraic manipulation is made on $[\,\boldsymbol{e}'\,]_\times = [\,\mathrm{P}'\boldsymbol{C}\,]_\times$.

Since $\boldsymbol{x}'$ lies on $\boldsymbol{l}'$ and $\boldsymbol{x}'^T\boldsymbol{l}' = \boldsymbol{0}$, which, substituting for A.14 can be written as

$$\boldsymbol{x}'^T\left(\mathrm{K}'^{-T}\,[\boldsymbol{t}]_\times\;\mathrm{R}\,\mathrm{K}^{-1}\right)\boldsymbol{x} = \boldsymbol{0} \tag{A.16}$$

This defines the epipolar geometry of the stereo system, where the fundamental matrix is given by

$$\mathrm{F} = \mathrm{K}'^{-T}\,[\boldsymbol{t}]_\times\;\mathrm{R}\,\mathrm{K}^{-1} \tag{A.17}$$

Computation of the fundamental matrix allows for the following calculation of the epipolar lines $\boldsymbol{l}' = \mathrm{F}\,\boldsymbol{x}$ and $\boldsymbol{l} = \mathrm{F}^T\,\boldsymbol{x}'$, since $\boldsymbol{x}^T\boldsymbol{l}' = \boldsymbol{0}$ and $\boldsymbol{l}^T\boldsymbol{x} = \boldsymbol{0}$ and the epipoles $\mathrm{F}\,\boldsymbol{e} = \boldsymbol{0}$ and $\mathrm{F}^T\boldsymbol{e}' = \boldsymbol{0}$.

The two cameras are related by a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$, collectively given as $[\,\mathrm{R}\,|\,\boldsymbol{t}\,]$, such that

$$\boldsymbol{x}' = \mathrm{R}\,\boldsymbol{x} + \boldsymbol{t} \tag{A.18}$$

Taking the vector product with $\boldsymbol{t}$, followed by the scalar product with $\boldsymbol{x}'$ yields,

$$\boldsymbol{x}' \cdot (\boldsymbol{t} \times \mathrm{R}\,\boldsymbol{x}) = \boldsymbol{0}$$

which returns to Equations A.9.

If the cameras are calibrated then the image point, in pixels, can be expressed in the camera coordinate system such that $\boldsymbol{x} = \mathrm{K}\,\boldsymbol{x}_c$ and $\boldsymbol{x}' = \mathrm{K}'\,\boldsymbol{x}'_c$. Substituting these and $\mathrm{E} = [\,\boldsymbol{t}\,]_\times\ \mathrm{R}$, where E is known as the *Essential* matrix, in Equation A.16 yields

$$\boldsymbol{x}_c'^T \mathrm{E}\,\boldsymbol{x}_c = \boldsymbol{0} \tag{A.19}$$

Comparing with Equation A.17, the essential matrix is related to the fundamental matrix by

$$\mathrm{F} = \mathrm{K}'^{-T}\,\mathrm{E}\,\mathrm{K}^{-1} \tag{A.20}$$

and defines the epipolar geometry for a stereo system with calibrated cameras.

Appendix B

People and Projects in 3D Scene Reconstruction

Some of the people associated with the three main projects at SMILE98 :

- **VANGUARD**: Visualization Across Networks using Graphics and Uncalibrated Acquisition of Real Data.
  European project based at the Visual Geometry Lab., Uni.of Oxford.
  P. A. Beardsley, G. Cross, A. Fitzgibbon, D. Forsyth, R. Koch, J. Mundy, M. Pollefeys, P. Pritchett, I. Reid, P. Torr, L. Van Gool, and A. Zisserman.
- **PANORAMA**: ACTS PANORAMA Project.
  Consortium of 14 European partners from various universities, research institutes and industry.
  R. Buschmann, L. Falkenhagen, R. Koch, A. Kopernik, T. Riegel, and D. Tzovaras.
- **CUMULI**: Computational Understanding of Multiple Images.
  European project with, INRIA Sophia-Antipolis, Lund Uni., INRIA-IMAG Grenoble, Fraunhofer IGD, IMetric.
  K. Åström, O. Faugeras, A. Heyden, R. Mohr, L. Quan, G. Sparr, P. Sturm, B. Triggs, and Z. Zhang.

A comprehensive list of poeple in Computer Vision has been compiled by Margaret Fleck at the and can be found at: `http://www.cs.hmc.edu/ fleck/computer-vision-handbook/vision-people.html`
Other projects of interest are:

- **CAMERA**: CAd Modelling of Built Environments from Range Analysis.
  Euro Project partners are: Fraunhofer IGD (Germany), IST (Portugal), IEC-JRC (Italy), Kungliga Tekniska Hogskolan (Sweden), LAAS-CNRS (France), UK Robotics (UK) and Uni.of Edinburgh (UK).
  Project Coordinator: R. Fisher, Uni.of Edingburgh.
  Aim: Automatically create architectural CAD and VR models of existing buildings from range and in-

tensity images.
Website: `http://www.dai.ed.ac.uk/daidb/people/homes/rbf/CAMERA/`

- **RECCAD**: (Copernicus) 3D surface Reconstruction for CAD modelling:
  European Research project partners are: VAGG, CS at Uni.of Cardiff., GML, CAR Inst. Budapest., CVL, Uni.of Ljubliana, Slovenia., CMP at Fac.Eng, Czech Tech.Uni.
  Aim: Produce complete and detailed boundary representations of geometric models.
  Website: `http://ralph.cs.cf.ac.uk/reccad.html`
- **COMORI and REC3D**: Construction of Complete 3-D Models from Range Images and a MathLab Toolbox for 3D Reconstruction from Uncalibrated 2D Views.
  CMP at Fac.Eng, Czech Tech.Uni.
  T. Werner, T. Pajdla, M. Urban, J. Burianek, J. Cernik and V. Hlavac
  Aim: Develop existing methods for registration of surfaces and fusion of surface measurements into consistent geometrical models.
  Website: `http://cmp.felk.cvut.cz/cmp/demos/Recx.html`
- **RESOLV**: Reconstruction using Scanned Laser and Video.
  European research programme, partners are; VERS Assoc., IST, JRC, Robosoft, ZGDV and Comp.Vis.Group, SCS at Uni.of Leeds.
  Project Manager: David Leevers of VERS.
  Aim: Full environment reconstruction by an autonomous mobile robot for telepresence applications, specifically surveying hazardous environments. Website: `http://www.scs.leeds.ac.uk/resolv/`
- **REVEAL**: Reconstruction from Video of Environments with Accurate Lighting.
  Advanced Interfaces Group, Dept.of CS at Uni.of Manchester.
  R. J. Hubbold, T. L. J. Howard, A. D. Murta, S. Gibson, A. J. West, D. Oram and J. Sinnott
  Aim: Construction of fully interactive virtual environments that faithfully represent real-world scenes.
  Website: `http://aig.cs.man.ac.uk/research/reveal/`
- **Temporal Stereo**: Electronic and Electrical Engineering Dept., at The University of Sheffield. S. Crossley, A. J. Lacey, R. A. Lane, N. L. Seed, N. A. Thacker and R. B. Yates
  Aim: Improving the accuracy and consistency of depth information for iterative 3D reconstruction using previous estimates of disparity images.
  Website: `http://www.shef.ac.uk/eee/ecs/index.html`
- **VECTOR**: Model-based Visual Surveillance.
  Computer Vision Group, CS at Uni.of Reading.
  G. Sullivan, S. Maybank, T. Tan, P. Remagnino, A. Worrall, J. Ferryman and J. Anderson
  Aim: Tracking of vehicles and people in urban scenes for security purposes and reconstruction of room interiors.
  Website: `http://www.cvg.cs.reading.ac.uk/Research.html`
- **VIRTUOUS**: Autonomous Virtual World construction.
  European research programme, partners are; CVSSP, EE at Uni.of Surrey., Inst.CTR Slovak Acad., UTIA CAS Prague IST Uni.of Lisbon.
  Aim: Registration of 3D surfaces to build 3D Models for a VR environment.
  Website: `http://www.ee.surrey.ac.uk/EE/VSSP/3DVision/virtuous/virtuous.html`

And in the US:

- **Facade**:
  Computer Vision Group, CS at Uni of California at Berkeley.
  P. Debevec, C. Taylor, J. Malik, G. Levin, Y. Yu
  Aim: Creation of photo-realistic models for the entertainment and the construction industries.
  Website: `http://www.cs.berkeley.edu/ debevec/Research/`
- **(MBV)** Modelling By Videotape:
  Robotics Inst. at Carnegie Mellon Uni.
  T. Kanade, C. Tomasi, C. J. Poelman, D. Moris, M. Han and L. Quan
  Aim: Autonomous 3D model construction via SFM.
  Website: `http://www.vasc.ri.cmu.edu/ mbv/`
- **Pioneer**: Remote reconnaissance system for structural analysis of the Chernobyl reactor.
  US Dept.of Energy lead consortium which includes NASA JPL and NREC.
  Aim: To design and build a tele-operated mobile robot capable of exploring the Chernobyl nuclear reactor and recovering detailed inspection of the interior.
  Website: `http://robotics.jpl.nasa.gov/tasks/pioneer/homepage.html`

Appendix C

General Considerations when Choosing a Framework for Image Processing Software for Computer Vision Research

Before any useful results can be obtained from research into image processing and/or computer vision techniques a software framework, or environment, for the tools being used, and/or developed, is essential. Even if this framework amounts to little more than an *ad-hoc* collection of processing and display programs, their interaction constitutes some form of framework and as such the following considerations will still apply.

The following describes, in brief, some of the considerations pertaining to an "informed" choice of a software environment, or framework, for image processing (IP) and computer vision (CV) tools. These considerations were applied to the choice of environment for work on large-scale 3D scene reconstruction but are equally valid for all areas of machine vision research. Use of the terms "framework" and "environment" are completely interchangeable in the context of this guide. It is hoped that this "alternative" guide, which is not intended as a review of frameworks currently available, will be of benefit to readers new to computer vision research and save valuable time for more important matters.

Acknowledgment: The author would like to thank his colleagues, Mike Lincoln and David Johnson, for their invaluable input to the formulation of ideas conveyed in this guide.

Off-the-Shelf Options - What to look for!

It seems that there are almost as many IP frameworks as there are applications. Much of this is due to the gradual evolution of CV techniques over the years and the increase in the number of machine vision applications but a great deal is also due to personal preference and the belief that an *in-house* system will be better in some way. In a few cases the later may be true but attempting to design a useful IP environment is by no means an easy task and can be a costly, albeit educational, exercise. There has been so much written about all the various IP environments that it is extremely difficult to offer a good reference from which to begin a review of the options available. However a useful collection of papers covering many of the important considerations can be found in 'Experimental Environments for Computer Vision and Image Processing [CC94a]'. A list of the most popular environments can be found through CVOnline together with an excellent review[1] by Dr Adrian Clark [Cla98], written originally for the EPSRC Summer School for Computer Vision. There are a number of similar reviews available on the 'Internet', some supporting specific environments, others more general. Therefore, yet another review is unnecessary. However, apart from the review referenced above, few, if any, give advice on what to look for and how to make an "informed" choice. The following hopes you address this issue without bias, so no judgments will be made about any specific frameworks. Regardless of

[1]also available from: `http://www.dai.ed.ac.uk/CVonline/environ.htm`

all the "hype" surrounding particular frameworks, none should be automatically regarded as the best option. In general, IP frameworks are initially written with a target area of research, or application, in mind and then adapted for more general use as they are developed. As such they may be ideally suited for the original task, but not necessarily ideal for others.

Image formats

The multitude of image formats[2] are a testimony to the complexity of the requirements of IP environments. At one end of the scale, a simple image format with just the height, width and number of bits per pixel is all that is required to carry out basic IP tasks. At the other, a comprehensive description of the data type, colour model, spectrum band, region of interest, history of processing and a whole host of other information may be required. There has been much debate over the year on the complexity of image formats and which ones an IP environment should support. However, a large proportion of the argument is ultimately dependent on the CV techniques being investigated. For example, work on multi-spectral (e.g. colour) images is greatly simplified if the data is stored in a single composite file rather than across several single band image files. However, processing is often performed on the separate image bands so efficient data management becomes an important issue. An image format with it's own compression might be good for data storage but adds a considerable overhead when attempting to process individual bands of data. There are a large number of format conversion tools freely available but most of the more popular environments already support an increasing number of image formats. It is advisable to become familiar with, at least, a couple of different image formats and choose the most appropriate for the work in hand. Try not to become preoccupied by one particular format. Saving and displaying results is paramount to all research work so choose image formats that are supported by a large number of processing, display and conversion tools.

[2]image format specifications can be found at: `http://www.dcs.ed.ac.uk/home/mxr/gfx/`

Component parts

If a software framework is considered as a set of co-operating “modules”, each can be analyzed in terms of its components. These modules can consist of any number of similar tools but there are essentially only a few distinct modules for certain types of software environment. These could be:

- Libraries of standard and/or task specific tools such as; mathematical, geometric, image processing, computer vision, simulation.
- A data management and process control core consisting of standard and user defined data types, data storage structures, memory management, process pipelining or scheduling, etc.
- User interfaces to provide data I/O, data visualization, which could include both 2D and 3D for images and data plots, and other analysis and prototyping tools to simplify algorithm development.

These types of “modules” have also been described as *toolkits* or *services* in some environments, but they amount to the same thing. Most, if not all, the popular environments have extensive libraries of the processing tools but, unfortunately, not all are organized in a user friendly way. Some of the larger environments are accompanied by list generating, or look-up, tools to help you find what you are looking, others seem to rely on poor descriptions and a considerable amount of user patience. Many environments also have extra toolkits for specific vision tasks, such as medical imaging or character recognition, but these are simply extensions to the basic processing libraries. “Ideally”, these more specialized CV modules should, in the option of the author, be optional and selected by compilation.

The core of any system is obviously fundamental to its operation and use. Most environments should be written to allow a considerable amount of flexibility for the access and manipulation of both image data and also the various sorts of dynamic data which are created from it, such as edges, corners or textures, etc. There are a number of data management models which regularly appear in IP environments, each with their own advantages and disadvantages. There is insufficient space here to discuss the various options in any detail but the reader should be aware of metaphors such as ‘stack’, ‘register’ and ‘blackboard’. The way in which an environment handles data processing internally may be of little interest to many users. However, such concerns become very apparent with increasing amounts of low-level programming within the environment. Of greater importance, at least to many, seems to be the type of user interface employed. This is understandable as there is little point in having an extensive library of processing tools if the effects can not be visualized in an appropriate way in order to realize their worth. However the effectiveness of user interfaces are often over shadowed by their appearance. Attractive features may appear to offer good functionality but in fact offer little of practical value. Ideally an IP framework should be able to display image data after every “distinct” process (*i.e.* those that change the image data). It should also support different methods of highlighting, or displaying, features of interest. This should be at various resolutions down to the pixel level. Some frameworks are able to support data plotting facilities, which also can be extremely useful. Tools which aid the construction of algorithms, either by scripted input or some form of graphical user interface, can be useful. However, they can also become hindrance to effective research. This is due to the human tendency to ignore essential checks of the derived data part way through the process chain. Discovering deep seated problems after months of looking at erroneous results is unfortunately more common that it should be. There is often considerable value attached to the *look and feel* of most software packages and unfortunately image processing environments appear to be no different.

In general, most of the popular IP environments have all the necessary components for an introduction into CV research. The choice of which to use therefore may not have been made easier by considering the component parts. If we assume that an IP environment has all the components required for research into the area of interest, then the next most important considerations are likely to be how easy they are to use and/or modify. Reviews of IP environments often discuss general attributes such as:

Extensibility - modify and extend to incorporate new ideas,

Generality - applicable to many areas of research by consisting of standardized tools and algorithms,

Operability - flexible interface that allows interaction at various levels of programming.

All these terms can have a number of different interpretations and are often used in impressive descriptions in the framework's accompanying documentation. Making an "informed" choice from such descriptions is not easy so it may be worth considering the choice of framework not by "what it can offer", but from the point of view of the actual requirement of the task in hand.

Level of programming

The concept of "requirement" introduces the most fundamental question that should be asked by all researchers connected with image processing and computer vision.

"What level of programming is required for the particular project or task".

If the level is more user oriented and less development then some high level interpretation, that allows the construction of algorithms by connecting together basic elements, may be more appropriate. This could involve relatively simple interfaces which consist of wrappers around lower level processes that are applied to images sequentially. Using such process wrappers, an edge detection algorithm might be developed by sequential calls to; `diffx()`, `sqr()`, `diffy()`, `sqr()`, `add()`, `nonmaxsup()`. Some environments use more exotic, higher level interfaces, some graphical and some scripted. The more popular environments allow different levels of programming access, but, unfortunately, others are more restrictive. The use of higher level, abstracted, user interfaces can increase understanding and productivity. However, such abstraction can also incur a considerable overhead when developing new code or testing new ideas, especially when manipulating data structures that were not expected to be handled by the particular interface.

If a considerable amount of code development is envisaged then access into the lower levels of source code will almost always be required. It seems that many researchers like to create their own, or at least modify existing, code. This includes both the IP tools and also the interface tools that produce the results in some particular format not normally available. Therefore, the level of programming access is often likely to be low level and involve direct access to all the source code. Many environments are freely available from the Internet, with full source code and documentation. Undoubtedly some of these will be written in a language which is preferred by the reader, however some may not. ‘C’ is a popular language for many mathematical and scientific software tools and there are a number language conversion packages available, both to and from ‘C’. There are also a number of possibilities of combining ‘C’ code with another languages. Although it is becoming increasingly less likely these days, some vision environments may contain large sections of code written in another language, or worse converted from some ancient library of low-level image processing functions. This is can lead to considerable problems when implementing new techniques and it is often difficult to confirm the suitability of the low-level processes in higher level algorithms. It is always advisable to conduct a thorough check of the source code before use.

The use of a particular programming language in IP and CV environments should be largely arbitrary. Unfortunately this is not always the case. The software models and structures within some languages, or, more accurately, the libraries of data classes and access functions created by some higher level languages, do not lend themselves to the sort of data access and manipulation required for IP and CV techniques. The greater the level of abstraction the more constrained the programmer is to using the predefined models and data structures. This can be apparent in both the design of the framework and the language it is written in. If the programming model at the required level of interaction is too rigid then it is often very difficult to implement new ideas.

In general, it is recommended that the choice of IP framework be based on the level of programming that will be undertaken. If this is at a high level, then a framework with good support tools for algorithm construction should be chosen. However, ensure that the tools are not too restrictive for code development. If, however, low level programming is envisaged, avoid high levels of abstraction, both in the code and the user interface.

Summary

The following summarizes the points raised above and can be used as a simple "checklist" for the choice of a framework for IP and CV tools. This should not be considered as absolute or complete.

- Image formats: Ensure they are appropriate for the project or task.
- Library support: Select the libraries you want and where possible leave out those you don't!
- System complexity: Ensure that there are appropriate image data types and dynamic data structures for the intended area of research and flexible, but consistent, forms of data access.
- Data visualization: Look for flexible image and data representation. This usually requires both 2D and 3D graphics support and plotting utilities.
- Productivity: Ensure that the framework design allows simple, quick algorithm generation but beware of "flashy" user interfaces.

Writing Your Own

Although there are some very good image processing environments, few, if any, of those which are freely available, offer all of the attributes that are required for efficient research. This is especially true when venturing into the higher levels of CV techniques such as those required for 3D scene reconstruction. Consequently, it is sometimes desirable to write your own. This should not be considered lightly. Designing and constructing a useful framework for vision research is not an easy task. However for those determined to embark on such an undertaking the following are a few pointers which were acquired from a couple of stalled attempts at constructing a dual image processing and virtual environment modelling framework for research use. This is not intended as a specification for a image processing framework and will not cover every aspect of possible concern. Neither will any aspects of the design of a framework for virtual environments be discussed.

JIVE (Joint Imaging and Virtual Environments) is an ongoing project at The University of Essex which hoped to combine many of the IP features identified above together with an interface for Virtual Reality (VR) modelling. Although this project had an ambitious specification, which incorporated multi-spectral images, video stream processing as well as a VR interface, the plan was to design and build the framework in two distinct phases. The first phase was just to construct the basic IP and CV framework with a versatile data management model that would allow the second, virtual environment, phase to be completed later. Unfortunately the project is still in its early stages of development but it is expected to be revitalized in the near future.

As identified earlier, an IP framework consists, in general terms, of three separate modules, a core, a set of processing libraries and some user interface tools. The following is considered in these terms in order to simplify the ideas. Few of these should actually be considered in isolation and most will impact on every component.

The basics

Definition of the internal image format needs to be flexible enough to handle almost anything. In practice this is almost impossible and some compromise must be made. Obvious data members for an image object, or class, are; height, width and a data pointer, ideally to a *void* array of pointers to rows in the image so that it is possible to define different data types to be used and allow maximum flexibility for data access. This immediately introduces the choice of language. Some languages are more strict on the specification and use of data types than others. This is both good and bad for the development of a framework which is usually designed to handle a number of different data types an efficient manner. Some languages are simply “non-starters” as far as image processing goes but that is a personal opinion so the reader is free to draw their own conclusions. There is also a question of the number of data types which should be supported. On the one hand in could be argued that “the more the better.” However, the greater the number of data types the greater the amount of code required to support each one. It is obvious that both integer and real data types are required and it is sensible to use the greatest possible precision for most numerical processes. It is recommended that both ‘complex’ and ‘pointer’ data types are also specified. The latter being particularly useful for the construction of images of pointers to structures such as ‘Edgel’ features which could contain useful, associated, data such as contrast, orientation, sub-pixel location, linkage and various other user-defined properties. From the point of view of data I/O, it is far better to design flexibility into the framework and allow for all possible data types, but then limit the internal processing to the types with the highest precision appropriate to the numerical process. However, unnecessary casting of data types is wasteful, especially if no useful processing takes place. Therefore the rigid definition of data types for library functions is should be generally avoided. A sensible balance can be drawn by using data types appropriate for the particular processing task being considered!

Most modern computer vision algorithms require full, random access to image data so a number of flexible methods are required. These method need to be consistent and ideally type invariant in order to simplify subsequent use. Apart form the usual “get-” and “put-line”, methods such as “get/put-column” and “get/put-pixel” are extremely useful. The management of dynamic structures is also very important and the use of various type of linked-list, including ‘tree’ and ‘graph’ structures is highly recommended. Few languages have built in error checking and those that do are unlikely to be versatile enough to deal with the requirements of an IP framework. Considering that most of the use this type of software will be developing new code, robust error handling should be designed into every level from the start. Thankfully, gone are the days when such frameworks were constrained to operate with very limited amounts of RAM so memory management can, in many cases, be left to the operating system. However, dedicated memory management can provide a useful method finding memory leaks and also avoiding runaway processes. The extra effort is probably worth the effort in the long term.

The inclusion of multi-spectra/multi-band (not necessarily just colour) images adds a new level of complexity to the problem of designing an IP framework. While a universal framework which treats all images the same is a desirable goal, the extra complexity it places on the system can be extremely restrictive. Historically, most multi-spectral processing were limited to single bands, effectively treated the same as grey-scale images, but advances in multi-spectral processing techniques, not least of which being *vector* processing, has lead to a need for much more flexible methods of data access. Apart from the obvious band access, full random access, down to pixel level, is often required. It may be tempting to consider incorporating multi-spectral processing at a low level of the core design, but this is not the best option. Although triple the amount (at best) of initial coding is required, there are considerable advantages to designing a separate core (or more accurately, part-core) module. Firstly it allows for multi-spectral processing to be “un-bolted” if not required by the user. Secondly, the fact that flexible data access is required suggests that decisions concerning internal storage of image data (band, line or pixel interleaved) need careful consideration. A balance between the abstraction of the image data and the full random access via, band, line and pixel is required that minimizes the amount on internal packing and re-packing of data. Such a requirement is difficult to implement in practice and would have a considerable impact on the performance of a single, standard core for grey-scale IP.

A further complication would be the inclusion of streamed image data. Again this will have considerable impact on the core, especially with memory management issues. It is recommended that if such a requirement is necessary it should be treated as a separate module and implemented through a specific interface to the core. The reason for this is similar to that given for multi-spectral images, but also stems from concerns about flexibility and speed. Unfortunately flexibility and speed are not compatible and one is almost always sacrificed to serve the other. If flexibility has already been designed into the core at a low level, then it seems counter productive to add constraints in order to service streamed data at that level, especially when a dedicated interface could be designed to handle the stream much more efficiently.

Finally, for this section, the use of *in-line* and/or macro functions can add considerable flexibility however the implementation can sometimes appear more complex than necessary. Ultimately the framework will be used by researchers with differing levels of programming ability so a number of options should be made available for the most widely used functions.

Libraries

Libraries of math, IP and CV tools can be borrowed from a number of different sources, not least of which is *Numerical Recipes* [PFTV93]. The use of borrowed libraries saves a considerable amount of coding but it is dangerous to assume that all the source code is valid. Such code should be tested as much as possible. Flexibility is always an important issue, so offering a number of alternative methods, especially those handling specific data types is recommended. However, trying to construct multiples of everything should be avoided. Some versions may never be required and if they are, then the user can generate their own from examples in the library. This last point leads to an extremely important issue concerned with compilation. If a framework is going of any use, the addition of new code must be quick and simple. This effectively means that regardless of the language used, compilation must be as simple as possible, yet allow for the addition of new, or replacement, elements. The use of *makefiles* often helps with such concerns but it is also necessary allow for preferential inclusion of source code files. Specification, or *header*, files should not however be much more difficult to over ride. These conflicting requirements are not always possible to implement in practice. Excessively long functions and/or files should be avoided. Break up large, or long, algorithms into sensible blocks. This is generally standard practice and allows for the maximum amount of code reuse. Always try and store similar code in the same place. Again the use of macro generated or in-line code is popular in many libraries but it is often difficult to find the definitions. Sensible distribution is recommended but the exact interpretation of this is left to the reader.

User interface tools and algorithm analysis

The creation of interface tools, which could include versatile display, data plot and algorithms analysis tools, for an IP framework can be the most time consuming part of the whole development. Creating good interactive tools is far more complicated than it first appears. For a start there is the obvious complication of dealing with event handling calls to the operating system's windowing functions. This is usually achieved by registering *call-back* functions for all of the *widgets* is use. Even something relatively simple can take many lines of code to implement. With so much code being devoted to the interface tools, it is often part of human nature that the developer would like their tools to be as attractive as possible. Unfortunately this often leads to a distraction from the functionality. It seems that for every good idea for a support tool and/or utility there are probably three bad ones. All researchers have their own interpretation of what constitutes a good tool and most would claim to be able to implement it better. The one fact that seems constant about support tools is that "the more they offer the worse they are". Attempting to write a tool that does everything will probably take twice as long, be ten times more complicated and be a hundred times less likely to be used. If "similar" functionality can be sensibly combined into a single tool then it should be included. If not, then a separate tool should be constructed. Again, sensible interpretation of this is required.

There are a few obvious tools that are required by all frameworks. Displaying results in an IP framework in paramount and the framework should be able to display the result after every stage of processing. This effectively means that all the supported image data types need to be converted into some common format for display. This undoubtedly requires different levels of truncation on each data type and could also include logarithmic re-scaling. Highlighting image data of interest with colours is common place, so the specification of local colour palette is fundamental. Camera style functions, such as zoom and pan, are particularly useful for image display in IP framework but obviously involve another level of complexity on top of basic system. Data plotting tools are extremely useful in IP frameworks but unfortunately there are very few, good, examples available. Apart from the usual 2D graphs and charts, 3D surface plots of image data are also useful but even harder to find. GNU have produced library of plotting utilities[3] which will aid development for unix/linux compatible framework. Algorithm analysis is often overlooked in many IP frameworks. The simple fact that the original author is happy with the performance of some process would seem to preclude the requirement for anyone else making a judgment. The liberal use of 'message' and 'data formatting' methods greatly aids the access of existing code as does and common text I/O interface. The idea of saving

[3]the library can be found at: `http://www.gnu.org/software/plotutils/plotutils.html`

comments to a console window which can then be edited and saved to file is particularly attractive. Processing timing is likely to be unreliable in a flexible IP framework, especially if the processing results are updated automatically (which is preferable), but the inclusion of a timer module, activated by simple ‘start’ and ‘stop’ calls, is often a useful guide when developing large algorithms with a number of component parts.

Design Strategy

A great deal of emphasis has been placed on the merits of object orientated (OO) design. In general, the methodology which OO supports is applicable to all software and should be applied and all times. However, while this is recommended for all commercial software, strict adherence to OO, by applying complete data abstraction, imposes excessive constraints on the research programmer. The need to ensure the stability of the lower level of a software framework is paramount. However, restricting access to higher levels of data, especially the image data and the algorithm derived from it, is counter productive. The researcher needs to be able to construct and manipulate data, at the outer most layer of the framework, with relative ease without incurring the penalty of an excessive number of calls to lower level functions. There is obviously a fine balance to be made between good object oriented design practice and over enthusiastic implementation of abstraction. The application of the *onion* model of the software system is a particularly useful metaphor for producing such a balance. In essence, decrease the level of data access from the outside layers inwards toward the centre.

Closure

The above are only a few observations and are not intended as a complete specification for the development of an image processing framework. The reader may use, or ignore, them as they see fit.

Appendix D

Code Listing for SUSAN Edge and Corner Detection

```
/* susan.h - */
#ifndef SUSAN
#define SUSAN

#define BLUTSCALE 100
#define SUSANMASK  37
#define EXPPWR      6
#define INTERCASE 600
#define ALMOST3   290
#define MINRESPNC  50
/*#define EDGE_RAW           0x00000000*/
#define EDGE_SUSAN         0x00000004

extern int *setup_blut(int **blutptr, int bth, int pwr);
extern void free_blut(int *blut);
extern void print_blut(int *blut);
extern Imrect *susan_prnc(Imrect *im, int geot, int *blut);
extern Imrect *susan_prnc2(Imrect *im, int geot, int *blut);
extern Imrect *susan_prnct(Imrect *im, int geot, int *blut);

extern Imrect *susan_corner(Imrect *im, float gthresh, int *blut);

extern Imrect *susan_edges(Imrect *im, float gthresh, float cfgmagn,
   float casecond, int *blut, int lengththres);

#endif

/* susan_prnc.c - This is the principle SUSAN algorithm used by
   both edge and corner detection procedures.
   Eddie Moxey, last update Oct 2000.
*/
#include <stdio.h>
#include <math.h>
/* tina headers */
#include <tina/sys.h>
#include <tina/sysfuncs.h>
```

```
#include <tina/math.h>
#include <tina/mathfuncs.h>
#include <tina/vision.h>
#include <tina/visionfuncs.h>
/* local headers */
#include "susan.h"
#include "../tina/eds_funcs.h"

static List *timed_proc_list = NULL;

int *setup_blut(int **blutptr, int bth, int pwr) {

  /* sets up a brightness look-up table for SUSAN algorithms */
  /* such that the similarity measure is given by:           */
  /*    sim(I, Io) =  exp((I - Io)/bth)^pwr                  */
  /*                 where bth is a brightness threshold.    */
  /* The values stored range from 0 (min) to 100 (max) only! */
  /* A pointer to the start of the LUT is returned but the   */
  /* original pointer given is used to access the LUT.       */
  int i;
  float tmp;
  int *blutfree;

  blutfree = ivector_alloc(0, 516);
  /* Only need 513 bytes for std 8-bit greyscale images */

  if(pwr == 0) pwr = EXPPWR;

  *blutptr = blutfree + 258; /* set pointer to centre to table */

  for(i = -256; i <= 256; i++) {

    tmp = ((float)i)/bth;
    tmp = tmp * tmp;       /* must be squared at least! */
    switch(pwr) {
    case 2 :
      break;
    case 4 :
      tmp = tmp * tmp;
      break;
    case 6 :
      tmp = tmp * tmp * tmp;
      break;
    default:
      error("Invalid exp power in SUSAN BLUT\n!", non_fatal);
    }
    tmp = BLUTSCALE * exp(-tmp);
    *(*blutptr + i) = (int)tmp;
  }
  return(blutfree);
}

void free_blut(int *blut) {

  ivector_free((void *)blut, 0);
}
void print_blut(int *blut) {
  /* test purposes only */
  int i, nout;
  for(i = -256; i <= 256; i++) {
    fprintf(stderr, "b%d %d\t", i, *(blut + i));
    if(!(nout = i % 8)) fprintf(stderr, "\n");
  }
}
void format_blut(int *blut) {
  /* test purposes only */
  int i, nout;
  for(i = -256; i <= 256; i++) {
    format("b%d %d\t", i, *(blut + i));
    if(!(nout = i % 8)) fprintf(stderr, "\n");
  }
}

Imrect* susan_prnc2(Imrect *im, int geot, int *blut) {
  /* susan principle 2 - as above but faster for corners  */
  /* because it constantly checks that the usan area does */
  /* not exceed the geot limit while incrementing.        */
  Imrect *im_out;
  int  *lineout, *im_lines[7];
  int  usanA;
  int  *centrep, *maskp;
  int  lx, ux, ly, uy;
  int  maskline, i, j, maski;

  if(im == NULL) {
    error("susan_prnc() given NULL image", non_fatal);
    return (NULL);
  }

  usanA = 0;
  lx = im->region->lx;
  ux = im->region->ux;
  ly = im->region->ly;
  uy = im->region->uy;

  im_out = im_alloc(im->height, im->width, im->region, int_v);
  lineout = ivector_alloc(lx, ux);

  for(maski = 0; maski < 7; maski++) {
    im_lines[maski] = ivector_alloc(lx, ux);
  }

  for(i = ly+5; i < uy-5; i++) {

    im_get_row(lineout, im_out, i, lx, ux);

    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {

im_get_row(im_lines[maskline], im, maski+i, lx, ux);
maskline++;
    }
    /* NB: 37 cell mask shape */
    /* is:  ..###..           */
    /*     .#####.           */
    /*      #######           */
    /*      ###X###           */
    /*      #######           */
    /*      .#####.           */
    /*      ..###..           */
    for(j = lx+5; j < ux-5; j++) {

      lineout[j] = 0;
      usanA = BLUTSCALE;  /* allow for central pixel now!*/
      centrep = blut + im_lines[3][j];
      /* define mask centre as an offset in the simlarity table */
      maskp = im_lines[0] + j - 1;
      /* Calculate USAN Area. */
      /* 1st line */
      usanA += *(centrep - *maskp++);
      /* subtract the pix value from the table offset */
      /* to find the similarity measure.              */
```

```
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp);
maskp = im_lines[1] + j - 2;  /* move on to next line */
/*2nd line */
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp);
maskp = im_lines[2] + j - 3;  /* move on to next line */
/* 3rd line */
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp);
maskp = im_lines[3] + j - 3;  /* move on to next line */
/* 4th and centre line */
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
usanA += *(centrep - *maskp++);
/* check usan area early and save computation */
if(usanA > geot) continue;
maskp ++;                     /* skip centre pixel */
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp);
if(usanA > geot) continue;
maskp = im_lines[4] + j - 3;  /* move on to next line */
/* 5th line */
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp);
if(usanA > geot) continue;
maskp = im_lines[5] + j - 2;  /* move on to next line */
/* 6th line */
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp);
if(usanA > geot) continue;
maskp = im_lines[6] + j - 1;  /* move on to next line */
/* 7th and last line */
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
usanA += *(centrep - *maskp++);
if(usanA > geot) continue;
```

```
      usanA += *(centrep - *maskp);

      if(usanA < geot) {
lineout[j] = (geot - usanA);
      }
    }
    /* end of row - put the mask lines back */
    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {

      im_put_row(im_lines[maskline], im, maski+i, lx, ux);
      maskline++;
    }
    im_put_row(lineout, im_out, i, lx, ux);
  }
  for(maski = 0; maski < 7; maski++) {
    ivector_free((void *)im_lines[maski], lx);
  }
  ivector_free((void *)lineout, lx);
  return (im_out);
}
/* End of File */

/* susan_edges.c
 * Smallest Univalue Segment Assimilating Nucleus(SUSAN)
 * Edge detection. Code is rather lengthy but efficient!
 * Eddie Moxey, May 2000.
*/

static Imrect *susan_edgmnts(Imrect *prncim, Imrect *im, int geot,
     float cfgmagn, float casecond, int* blut) {
  /* calc the usan area moments for subsequent edge orientation */
  Imrect   *maxim;
  int      cgx, cgy;
  long     cgxsq, cgysq, xycfg;
  long     sqsum, cfgvecmag;
  int      *prnclines[5], *im_lines[7];
  int      siml, usanA, usanR;
  int      *centrep, *maskp;
  int      lx, ux, ly, uy;
  int      maskline, i, j, maski, a, b, w;
  Bool     do_secndmnts = false;
  float    cheapatan, posoff, drow, dcol;
  Vec2     pos = {Vec2_id};
  Edgel    *eptr, *edge_alloc();

  if (im == NULL || prncim == NULL) {
    error("susan_edgmnts() given NULL image", non_fatal);
    return (NULL);
  }
  cgx = cgy = cgxsq = cgysq = sqsum = xycfg = 0;
  siml = usanA = usanR = 0;

  lx = im->region->lx;
  ux = im->region->ux;
  ly = im->region->ly;
  uy = im->region->uy;

  maxim = im_alloc(im->height, im->width, im->region, ptr_v);

  for(maski = 0; maski < 7; maski++) {
    im_lines[maski] = ivector_alloc(lx, ux);
  }
  for(maski = 0; maski < 3; maski++) {
    prnclines[maski] = ivector_alloc(lx, ux);
  }
  /* Only need to process the centre area of im,*/
  /* so allow some spare lines around the edge! */
  for(i = ly+7; i < uy-7; i++) {

    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {

      im_get_row(im_lines[maskline], im, maski+i, lx, ux);
      maskline++;
    }
    maskline = 0;
    for(maski = -1; maski <= 1; maski++) {
      im_get_row(prnclines[maskline], prncim, maski+i, lx, ux);
      maskline++;
    }
    for(j = lx+7; j < ux-7; j++) {

      usanR = prnclines[1][j];
      if(usanR < 50) continue;
      /* Skip if original edge no response! */
      usanA = geot - usanR;
      if(usanA < 200) continue;
      /* Skip if original usanA was too small !  */
      centrep = blut + im_lines[3][j];

      if(usanA > casecond) {
/* Calculate 1st Moments for Centre of Gravity */
/* of USAN area. casecond replaced INTERCASE */
cgx = cgy = 0;
maskp = im_lines[0] + j - 1;
/* 1st line */
siml = *(centrep - *maskp++);
cgx -= siml;    cgy -= 3*siml;
siml = *(centrep - *maskp++);
cgy -= 3*siml;
siml = *(centrep - *maskp);
cgx += siml;    cgy -= 3*siml;
maskp = im_lines[1] + j - 2;  /* move on to next line */
/* 2nd line */
siml = *(centrep - *maskp++);
cgx -= 2*siml;  cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;    cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgx += siml;    cgy -= 2*siml;
siml = *(centrep - *maskp);
cgx += 2*siml;  cgy -= 2*siml;
maskp = im_lines[2] + j - 3;  /* move on to next line */
/* 3rd line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;   cgy -= siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;   cgy -= siml;
siml = *(centrep - *maskp++);
cgx -= siml;     cgy -= siml;
siml = *(centrep - *maskp++);
cgy -= siml;
siml = *(centrep - *maskp++);
cgx += siml;     cgy -= siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;   cgy -= siml;
siml = *(centrep - *maskp);
cgx += 3*siml;   cgy -= siml;
maskp = im_lines[3] + j - 3;  /* move on to next line */
```

```
/* 4th line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;
maskp ++;         /* skip centre pixel */
siml = *(centrep - *maskp++);
cgx += siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;
siml = *(centrep - *maskp++);
cgx += 3*siml;
maskp = im_lines[4] + j - 3;  /* move on to next line */
/* 5th line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;   cgy += siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;   cgy += siml;
siml = *(centrep - *maskp++);
cgx -= siml;     cgy += siml;
siml = *(centrep - *maskp++);
cgy += siml;
siml = *(centrep - *maskp++);
cgx += siml;     cgy += siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;   cgy += siml;
siml = *(centrep - *maskp);
cgx += 3*siml;   cgy += siml;
maskp = im_lines[5] + j - 2;  /* move on to next line */
/* 6th line */
siml = *(centrep - *maskp++);
cgx -= 2*siml;  cgy += 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;    cgy += 2*siml;
siml = *(centrep - *maskp++);
cgy += 2*siml;
siml = *(centrep - *maskp++);
cgx += siml;    cgy += 2*siml;
siml = *(centrep - *maskp);
cgx += 2*siml;  cgy += 2*siml;
maskp = im_lines[6] + j - 1;  /* move on to next line */
/* 7th and last line */
siml = *(centrep - *maskp++);
cgx -= siml;    cgy += 3*siml;
siml = *(centrep - *maskp++);
cgy += 3*siml;
siml = *(centrep - *maskp);
cgx += siml;    cgy += 3*siml;

/* Compare abs distance of cofg from nucleus */
cgxsq = SQR(cgx);
cgysq = SQR(cgy);
sqsum = cgxsq + cgysq;
cfgvecmag = (long)(cfgmagn * SQR(usanA));

if( sqsum > cfgvecmag ) {
  /* i.e is cofg vector magnitude > 0.? * usanA  */
  /* If cofg NOT too close to nucleus then can   */
  /* calculate edgel with 1st moments.           */
  /* NB: Original range from 0.5 to 0.9 but      */
  /* this appears to be tight. Problem with      */
  /* edges at sharp corners where usuaA is small */
  do_secndmnts = false;
  if( cgx == 0 )
    /* must be horizontal */
    cheapatan = 100.0;  /* anything greater than 2.0 */
  else
    cheapatan = ((float)cgy)/((float)cgx);
  /* check which sector CofG is in */
  if( cheapatan < 0 ) {
    cheapatan = -cheapatan;
    w = -1;
  }
  else
    w = 1;
  /* each sector divided into: < 26.6 deg ( atan(0.5) )*/
  /* > 63.4 deg ( atan(2.0) ), or  26.6 > < 63.4 deg.  */
  /* These are the only sensible args to atan which    */
  /* give a symetrical division between vertical,      */
  /* horizontal or diagonal orientation  !!            */
  if( cheapatan < 0.5 ) { /* vertical edge, a=0,b=1 */
    /* do NMS across the edge then set sub-pix pos */
    a  = prnclines[1][j-1];
    b  = prnclines[1][j+1];
    if( (usanR < a) || (usanR <= b) )
      continue;

    posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
    /* 1D quadratic curve fit across the edge which does  */
    /* not give very accurate position for adjacent pixls */
    /* with the same initial response but will do for now.*/
    /* 2D quadratic surface fit would be better!          */
    drow = 0.5;
    dcol = posoff + 0.5;
  }
  else {
    if( cheapatan > 2.0) { /* horizontal edge, a=1,b=0*/
      /* NB: get pixels at 90 degree to edge! */
      a    = prnclines[0][j];
      b    = prnclines[2][j];
      if( (usanR < a) || (usanR <= b) )
continue;

      posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
      drow = posoff + 0.5;
      dcol = 0.5;
    }
    else { /* diagonal edge (0.5 < z < 2.0) */
      if ( w > 0 ) { /* -ve diagonal, a=1,b=1*/
/* NB: edge in line with diag cofg vector */
a    = prnclines[0][j-1];
b    = prnclines[2][j+1];
if( (usanR < a) || (usanR <= b) )
  continue;

posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
drow = posoff + 0.5;
dcol = posoff + 0.5;
      }
      else { /* -ve W but +ve diagonal, a=-1,b=1 */
a    = prnclines[2][j-1];
b    = prnclines[0][j+1];  /* swapped here! */
if( (usanR <= a) || (usanR < b) )
  continue;

posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
drow = 0.5 - posoff;
dcol = posoff + 0.5;
      }
    }
  }
  pos = vec2(j + dcol, i + drow);
  eptr = edge_alloc(EDGE_SUSAN);
  eptr->pos = pos;
  eptr->contrast = (float)usanR;

  /* determine edge orientation */
  if( ABS(cgx) < MINRESPNC ) {
    /* cofg is nearly vertical */
    if( cgy < 0 )
      eptr->orient = PI;
    else
      eptr->orient = 0;
  }
  else if( ABS(cgy) < MINRESPNC ) {
    /* cofg is nearly horizontal */
    if( cgx < 0 )
      eptr->orient = PIBY2;
    else
      eptr->orient = -PIBY2;
  }
  else {
    if ( SAME_SIGN(cgx, cgy) )
      eptr->orient = atan2(CHX_SIGN((double)cgy), (double)cgx);
    else
      eptr->orient = atan2((double)cgy, CHX_SIGN((double)cgx));
  }
  IM_PTR(maxim, i, j) = (void *)eptr;
}
else {
  do_secndmnts = true;
        }
      } /* end of - if(usanA > INTERCASE) */
      else /* Must be small USAN Area, 1st momnts no good */
        do_secndmnts = true;

      if( do_secndmnts ) {

cgxsq = cgysq = xycfg = 0;
/* Calculate 2nd Moments */
maskp = im_lines[0] + j - 1;
/* 1st line */
siml = *(centrep - *maskp++);
cgxsq += siml;   cgysq += 9*siml;   xycfg += 3*siml;
siml = *(centrep - *maskp++);
cgysq += 9*siml;
siml = *(centrep - *maskp);
cgxsq += siml;   cgysq += 9*siml;   xycfg -= 3*siml;
maskp = im_lines[1] + j - 2;  /* move on to next line */
/* 2nd line */
siml = *(centrep - *maskp++);
cgxsq += 4*siml;  cgysq += 4*siml;  xycfg += 4*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;    cgysq += 4*siml;  xycfg += 2*siml;
siml = *(centrep - *maskp++);
cgysq += 4*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;    cgysq += 4*siml;  xycfg -= 2*siml;
siml = *(centrep - *maskp);
cgxsq += 4*siml;  cgysq += 4*siml;  xycfg -= 4*siml;
maskp = im_lines[2] + j - 3;  /* move on to next line */
/* 3rd line */
siml = *(centrep - *maskp++);
cgxsq += 9*siml;   cgysq += siml;   xycfg += 3*siml;
siml = *(centrep - *maskp++);
```

```
cgxsq += 4*siml;   cgysq += siml;   xycfg += 2*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;     cgysq += siml;   xycfg += siml;
siml = *(centrep - *maskp++);
cgysq += siml;
siml = *(centrep - *maskp++);
cgxsq += siml;     cgysq += siml;   xycfg -= siml;
siml = *(centrep - *maskp++);
cgxsq += 4*siml;   cgysq += siml;   xycfg -= 2*siml;
siml = *(centrep - *maskp);
cgxsq += 9*siml;   cgysq += siml;   xycfg -= 3*siml;
maskp = im_lines[3] + j - 3;  /* move on to next line */
/* 4th line */
siml = *(centrep - *maskp++);
cgxsq += 9*siml;
siml = *(centrep - *maskp++);
cgxsq += 4*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;
maskp ++;         /* skip centre pixel */
siml = *(centrep - *maskp++);
cgxsq += siml;
siml = *(centrep - *maskp++);
cgxsq += 4*siml;
siml = *(centrep - *maskp++);
cgxsq += 9*siml;
maskp = im_lines[4] + j - 3;  /* move on to next line */
/* 5th line */
siml = *(centrep - *maskp++);
cgxsq += 9*siml;   cgysq += siml;   xycfg -= 3*siml;
siml = *(centrep - *maskp++);
cgxsq += 4*siml;   cgysq += siml;   xycfg -= 2*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;     cgysq += siml;   xycfg -= siml;
siml = *(centrep - *maskp++);
cgysq += siml;
siml = *(centrep - *maskp++);
cgxsq += siml;     cgysq += siml;   xycfg += siml;
siml = *(centrep - *maskp++);
cgxsq += 4*siml;   cgysq += siml;   xycfg += 2*siml;
siml = *(centrep - *maskp);
cgxsq += 9*siml;   cgysq += siml;   xycfg += 3*siml;
maskp = im_lines[5] + j - 2;  /* move on to next line */
/* 6th line */
siml = *(centrep - *maskp++);
cgxsq += 4*siml;  cgysq += 4*siml;  xycfg -= 4*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;    cgysq += 4*siml;  xycfg -= 2*siml;
siml = *(centrep - *maskp++);
cgysq += 4*siml;
siml = *(centrep - *maskp++);
cgxsq += siml;    cgysq += 4*siml;  xycfg += 2*siml;
siml = *(centrep - *maskp);
cgxsq += 4*siml;  cgysq += 4*siml;  xycfg += 4*siml;
maskp = im_lines[6] + j - 1;  /* move on to next line */
/* 7th and last line */
siml = *(centrep - *maskp++);
cgxsq += siml;    cgysq += 9*siml;  xycfg -= 3*siml;
siml = *(centrep - *maskp++);
cgysq += 9*siml;
siml = *(centrep - *maskp);
cgxsq += siml;    cgysq += 9*siml;  xycfg += 3*siml;

if( cgysq == 0 )
  /* must be horizontal line, 1 pixel wide */
  cheapatan = 100.0;  /* anything greater than 2.0 */
```

```
else
  cheapatan = ((float)cgxsq)/((float)cgysq);

if( cheapatan < 0.5 ) { /* vertical edge */
  a    = prnclines[1][j-1];
  b    = prnclines[1][j+1];
  if( (usanR < a) || (usanR <= b) )
    continue;
  posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
  drow = 0.5;
  dcol = posoff + 0.5;
}
else {
  if( cheapatan > 2.0) { /* horizontal edge */
    a    = prnclines[0][j];
    b    = prnclines[2][j];
    if( (usanR < a) || (usanR <= b) )
      continue;

    posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
    drow = posoff + 0.5;
    dcol = 0.5;
  }
  else { /* diagonal edge */
    if ( xycfg > 0 ) { /* -ve diagonal, a=-1,b=1 */
      a    = prnclines[2][j-1];
      b    = prnclines[0][j+1];
      if( (usanR <= a) || (usanR < b) )
continue;

      posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
      drow = 0.5 - posoff;
      dcol = posoff + 0.5;
    }
    else { /* +ve diagonal, a=1,b=1 */
      a    = prnclines[0][j-1];
      b    = prnclines[2][j+1];
      if( (usanR < a) || (usanR <= b) )
continue;
      posoff = (float)(a - b)/(2 * (a + b - 2*usanR));
      drow = posoff + 0.5;
      dcol = posoff + 0.5;
    }
  }
}
pos = vec2(j + dcol, i + drow);
eptr = edge_alloc(EDGE_SUSAN);
eptr->pos = pos;
eptr->contrast = (float)usanR;

if( cgxsq < MINRESPNC )
  eptr->orient = PIBY2;
else if( cgysq < MINRESPNC )
  eptr->orient = 0;
else {
  if( xycfg == 0 ) {
    if( (cgxsq - cgysq) <= 0 )
      eptr->orient = PIBY2;
    else
      eptr->orient = 0;
  }
  else
    eptr->orient = 0.5 * asin(xycfg/
      sqrt(SQR(xycfg) + SQR(cgxsq-cgysq)));
}
IM_PTR(maxim, i, j) = (void *)eptr;
```

```
      }
    }
    maskline = 0;
    for(maski = -1; maski <= 1; maski++) {
      im_put_row(prnclines[maskline], prncim, maski+i, lx, ux);
      maskline++;
    }
    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {
      im_put_row(im_lines[maskline], im, maski+i, lx, ux);
      maskline++;
    }
  }
  for(maski = 0; maski < 3; maski++) {
    ivector_free((void *)prnclines[maski], lx);
  }
  for(maski = 0; maski < 7; maski++) {
    ivector_free((void *)im_lines[maski], lx);
  }
  return (maxim);
}

Imrect* susan_edges(Imrect* im, float geothresh, float cfgmagn,
    float casecond, int *blut, int lengththres) {

  Imrect *prnc_im, *edge_im;
  int    lx, ux, ly, uy;
  int    gthresh = 0;

  if (im == NULL) return (NULL)
  lx = im->region->lx;
  ux = im->region->ux;
  ly = im->region->ly;
  uy = im->region->uy;

  gthresh = (int)(SUSANMASK * BLUTSCALE * geothresh - BLUTSCALE);
  /*(37 x 100 x 0.75)-100 = 2675, original magic num was 2650 */

  /* do susan principle pass */
  prnc_im = susan_prnc(im, gthresh, blut);

  edge_im = susan_edgmnts(prnc_im, im, gthresh, cfgmagn, casecond, blut);
  er_find_edge_strings(edge_im);
  /**er_find_simple_edge_strings(edge_im);**/
  er_rm_edges(edge_im, EDGE_GET_CONN_MASK, EDGE_NOLINK);
  /**er_edge_strings_thres(edge_im, lengththres, upthres);NOT REQ’D!**/
  er_set_row_index(edge_im);

  im_free(prnc_im);
  return(edge_im);
}
/* EOF */

/* susan_crns.c
   Corner detection using the SUSAN algorithm.
   Eddie Moxey, version 2.0, updated October 2000
*/

static Imrect *susan_cnrmnts(Imrect *prncim, Imrect *im,
     int geot, int* blut,
     Imrect **cofgx, Imrect **cofgy) {
  /* calc the usan area moments and c/o the centroid and */
  /* contiguity tests for true corners */
  Imrect *im_out;
  int  *lineout, *cgxline, *cgyline;
  int  cgx, cgy, cgxsq, cgysq, sqsum, cofgn;
  float cofgd = 0.0;
  int  *prncline, *im_lines[7];
  int  siml, usanR;
  int  *centrep, *maskp;
  int  lx, ux, ly, uy;
  int  maskline, i, j, maski;

  if (im == NULL) {
    error("susan_cnrmnts() given NULL image", non_fatal);
    return (NULL);
  }
  cgx = cgy = cgxsq = cgysq = sqsum = 0;
  siml = usanR = 0;

  lx = im->region->lx;
  ux = im->region->ux;
  ly = im->region->ly;
  uy = im->region->uy;

  im_out = im_alloc(im->height, im->width, im->region, int_v);
  lineout = ivector_alloc(lx, ux);
  prncline = ivector_alloc(lx, ux);
  cgxline = ivector_alloc(lx, ux);
  cgyline = ivector_alloc(lx, ux);

  for(maski = 0; maski < 7; maski++) {
    im_lines[maski] = ivector_alloc(lx, ux);
  }
  /* No need to process around the edge of image. */
  for(i = ly+5; i < uy-5; i++) {

    im_get_row(prncline, prncim, i, lx, ux);
    im_get_row(cgxline, *cofgx, i, lx, ux);
    im_get_row(cgyline, *cofgy, i, lx, ux);
    im_get_row(lineout, im_out, i, lx, ux);

    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {
im_get_row(im_lines[maskline], im, maski+i, lx, ux);
maskline++;
    }
    for(j = lx+5; j < ux-5; j++) {
      /* NB: Zero outputs first! */
      lineout[j] = cgxline[j] = cgyline[j] = 0;

      usanR = prncline[j];
      /* Check principle response for possible corner */
      /* NB: could be from combined principle response! */
      if( usanR < 200 ) continue;
      /* Check for noise pixels - V large org response! */
      if( usanR > (geot - 200) ) continue;

      /* Calculate 1st and 2nd Moments */
      cgx = cgy = 0;
      centrep = blut + im_lines[3][j];
      maskp = im_lines[0] + j - 1;
      /* 1st line */
      siml = *(centrep - *maskp++);
      cgx -= siml;    cgy -= 3*siml; /* inc cgw here ?? */
      siml = *(centrep - *maskp++);
      cgy -= 3*siml;
      siml = *(centrep - *maskp);
      cgx += siml;    cgy -= 3*siml;
      maskp = im_lines[1] + j - 2;  /* move on to next line */
      /* 2nd line */
      siml = *(centrep - *maskp++);
```

```
cgx -= 2*siml;  cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;    cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgy -= 2*siml;
siml = *(centrep - *maskp++);
cgx += siml;    cgy -= 2*siml;
siml = *(centrep - *maskp);
cgx += 2*siml;  cgy -= 2*siml;
maskp = im_lines[2] + j - 3;  /* move on to next line */
/* 3rd line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;   cgy -= siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;   cgy -= siml;
siml = *(centrep - *maskp++);
cgx -= siml;     cgy -= siml;
siml = *(centrep - *maskp++);
cgy -= siml;
siml = *(centrep - *maskp++);
cgx += siml;     cgy -= siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;   cgy -= siml;
siml = *(centrep - *maskp);
cgx += 3*siml;   cgy -= siml;
maskp = im_lines[3] + j - 3;  /* move on to next line */
/* 4th line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;
maskp ++;         /* skip centre pixel */
siml = *(centrep - *maskp++);
cgx += siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;
siml = *(centrep - *maskp++);
cgx += 3*siml;
maskp = im_lines[4] + j - 3;  /* move on to next line */
/* 5th line */
siml = *(centrep - *maskp++);
cgx -= 3*siml;   cgy += siml;
siml = *(centrep - *maskp++);
cgx -= 2*siml;   cgy += siml;
siml = *(centrep - *maskp++);
cgx -= siml;     cgy += siml;
siml = *(centrep - *maskp++);
cgy += siml;
siml = *(centrep - *maskp++);
cgx += siml;     cgy += siml;
siml = *(centrep - *maskp++);
cgx += 2*siml;   cgy += siml;
siml = *(centrep - *maskp);
cgx += 3*siml;   cgy += siml;
maskp = im_lines[5] + j - 2;  /* move on to next line */
/* 6th line */
siml = *(centrep - *maskp++);
cgx -= 2*siml;  cgy += 2*siml;
siml = *(centrep - *maskp++);
cgx -= siml;    cgy += 2*siml;
siml = *(centrep - *maskp++);
cgy += 2*siml;
siml = *(centrep - *maskp++);
cgx += siml;    cgy += 2*siml;
```

```
      siml = *(centrep - *maskp);
      cgx += 2*siml;  cgy += 2*siml;
      maskp = im_lines[6] + j - 1;  /* move on to next line */
      /* 7th and last line */
      siml = *(centrep - *maskp++);
      cgx -= siml;    cgy += 3*siml;
      siml = *(centrep - *maskp++);
      cgy += 3*siml;
      siml = *(centrep - *maskp);
      cgx += siml;    cgy += 3*siml;

      /* Compare abs distance of cofg from nucleus */
      cgxsq = SQR(cgx);
      cgysq = SQR(cgy);
      sqsum = cgxsq + cgysq;

      if( (float)sqsum > (0.5*SQR(geot-usanR)) ) {
/* i.e is cofg vector magnitude > 0.707 * z*/
/* If cofg NOT too close to nucleus then   */
/* must be a corner - So Check contiguity  */
if(cgysq < cgxsq) {
  /* More Lateral */
  cofgd = (float)cgy/ABS(cgx);
  cofgn = ABS(cgx)/cgx;        /* normalise +/-1 */
  sqsum = *(centrep -
    *(im_lines[3+FTOI(cofgd)] + j+cofgn)) +
    *(centrep - *(im_lines[3+FTOI(2*cofgd)] + j+2*cofgn)) +
    *(centrep - *(im_lines[3+FTOI(3*cofgd)] + j+3*cofgn));
  /* reuse sqsum as straight line strength */
}
else {
  /* More Longitudal */
  cofgd = (float)cgx/ABS(cgy);
  cofgn = ABS(cgy)/cgy;
  sqsum = *(centrep -
    *(im_lines[3+cofgn] + j+FTOI(cofgd))) +
    *(centrep - *(im_lines[3+2*cofgn] + j+FTOI(2*cofgd))) +
    *(centrep - *(im_lines[3+3*cofgn] + j+FTOI(3*cofgd)));
}
if(sqsum > ALMOST3) {
  /* To be a true corner 3 pixels in usan must be in */
  /* a straight line radiating out from the nucleus */
  lineout[j] = prncline[j];
  cgxline[j] = cgx;
  cgyline[j] = cgy;
}
      }
    }
    maskline = 0;
    for(maski = -3; maski <= 3; maski++) {

      im_put_row(im_lines[maskline], im, maski+i, lx, ux);
      maskline++;
    }
    im_put_row(prncline, prncim, i, lx, ux);
    im_put_row(cgxline, *cofgx, i, lx, ux);
    im_put_row(cgyline, *cofgy, i, lx, ux);
    im_put_row(lineout, im_out, i, lx, ux);
  }
  for(maski = 0; maski < 7; maski++) {
    ivector_free((void *)im_lines[maski], lx);
  }
  ivector_free((void *)prncline, lx);
  ivector_free((void *)cgxline, lx);
  ivector_free((void *)cgyline, lx);
  ivector_free((void *)lineout, lx);
  return(im_out);
}

static Imrect* check_cornim(Imrect *cornim) {
  /* Test function only! */
  Imrect  *max_im;
  int     pix;
  int     ncnrs = 0;
  int     lx, ux, ly, uy;
  int     i, j;
  Edgel   *eptr;

  if (cornim == NULL) return (NULL);

  lx = cornim->region->lx;
  ux = cornim->region->ux;
  ly = cornim->region->ly;
  uy = cornim->region->uy;

  max_im = im_alloc(cornim->height, cornim->width, cornim->region, ptr_v);

  for(i = ly+5; i < uy-5; i++) {
    for(j = lx+5; j < ux-5; j++) {

      if( (pix = im_get_pix(cornim, i, j)) < 200 ) continue;
      /* Only process non-zero responses; */

      eptr = edge_alloc(EDGE_SUSAN);
      eptr->contrast = (float)pix;
      eptr->pos = vec2(j + 0.5, i + 0.5);
      eptr->orient = PI;
      eptr->type &= EDGE_SET_CONN_MASK;
      eptr->type |= EDGE_ISOLATED;

      IM_PTR(max_im, i, j) = (void *)eptr;
    }
  }
  return(max_im);
}

static Imrect* susan_locatcnr(Imrect *cornim, Imrect *prncim,
      Imrect *cofgx, Imrect *cofgy) {

  Imrect  *max_im;
  int     *masklines[5];
  int     *cgxline, *cgyline;
  int     cgx, cgy, pix;
  float   dx, dy;
  Vec2    pos = {Vec2_id};
  int     lx, ux, ly, uy;
  int     maskline, i, j, maski;
  Edgel   *eptr;

  if ((cornim == NULL)||(prncim == NULL)) return (NULL);

  lx = cornim->region->lx;
  ux = cornim->region->ux;
  ly = cornim->region->ly;
  uy = cornim->region->uy;

  max_im = im_alloc(cornim->height, cornim->width, cornim->region, ptr_v);
  cgxline = ivector_alloc(lx, ux);
  cgyline = ivector_alloc(lx, ux);

  for(maski = 0; maski < 5; maski++) {
    /* use a 5x5 mask to identify centre of smoothed corner profile, */
    /* i.e. centre of a banded edge! */
    masklines[maski] = ivector_alloc(lx, ux);
  }

  for(i = ly+7; i < uy-7; i++) {
    im_get_row(cgxline, cofgx, i, lx, ux);
    im_get_row(cgyline, cofgy, i, lx, ux);

    maskline = 0;
    for(maski = -2; maski <= 2; maski++) {
      /* use 5x5 mask to isolate from other corners near by!*/
      im_get_row(masklines[maskline], cornim, maski+i, lx, ux);
      maskline++;
    }

    for(j = lx+7; j < ux-7; j++) {

      if((pix = masklines[2][j]) < 200) continue;
      /* Only process non-zero responses; */
      /* find strongest corner */
      if(pix < *(masklines[0] + j-2)) continue;
      if(pix < *(masklines[0] + j-1)) continue;
      if(pix < *(masklines[0] + j  )) continue;
      if(pix < *(masklines[0] + j+1)) continue;
      if(pix < *(masklines[0] + j+2)) continue;
      /* 2nd line */
      if(pix < *(masklines[1] + j-2)) continue;
      if(pix < *(masklines[1] + j-1)) continue;
      if(pix < *(masklines[1] + j  )) continue;
      if(pix < *(masklines[1] + j+1)) continue;
      if(pix < *(masklines[1] + j+2)) continue;
      /* 3rd line */
      if(pix < *(masklines[2] + j-2)) continue;
      if(pix < *(masklines[2] + j-1)) continue;
      /* skip centre */
      if(pix < *(masklines[2] + j+1)) continue;
      if(pix < *(masklines[2] + j+2)) continue;
      /* 4th line */
      if(pix < *(masklines[3] + j-2)) continue;
      if(pix < *(masklines[3] + j-1)) continue;
      if(pix < *(masklines[3] + j  )) continue;
      if(pix < *(masklines[3] + j+1)) continue;
      if(pix < *(masklines[3] + j+2)) continue;
      /* last line */
      if(pix < *(masklines[4] + j-2)) continue;
      if(pix < *(masklines[4] + j-1)) continue;
      if(pix < *(masklines[4] + j  )) continue;
      if(pix < *(masklines[4] + j+1)) continue;
      if(pix < *(masklines[4] + j+2)) continue;
      /* pix must be the local max */

      eptr = edge_alloc(EDGE_SUSAN);
      eptr->contrast =
im_get_quadmaxi(prncim,(float)j,(float)i,&dx,&dy);
      /*new func from  ./imintfns.c */
      eptr->pos = vec2(dx,dy);
      eptr->type &= EDGE_SET_CONN_MASK;
      eptr->type |= EDGE_ISOLATED;

      cgx = cgxline[j];
      cgy = cgyline[j];

      if( ABS(cgx) < MINRESPNC ) {
/* cofg is nearly vertical */
if( cgy < 0 )
  eptr->orient = PIBY2;
else
  eptr->orient = -PIBY2;
      }
      else if( ABS(cgy) < MINRESPNC ) {
/* cofg is nearly horizontal */
if( cgx < 0 )
  eptr->orient = 0;
else
  eptr->orient = PI;
      }
      else {
eptr->orient = atan2(CHX_SIGN((double)cgy),
                     CHX_SIGN((double)cgx) );
      }
      IM_PTR(max_im, i, j) = (void *)eptr;
    }
    maskline = 0;
    for(maski = -2; maski <= 2; maski++) {
      im_put_row(masklines[maskline], cornim, maski+i, lx, ux);
      maskline++;
    }
  }
  for(maski = 0; maski < 5; maski++) {
    ivector_free((void *)masklines[maski], lx);
  }
  ivector_free((void *)cgxline, lx);
  ivector_free((void *)cgyline, lx);
  return(max_im);
}

Imrect* susan_corner(Imrect *im, float geothresh, int *blut) {

  Imrect *corn_im, *prnc_im;
  Imrect *cofgx, *cofgy, *maxim;
  int    lx, ux, ly, uy;
  int    gthresh = 0;

  if (im == NULL) return (NULL);

  lx = im->region->lx;
  ux = im->region->ux;
  ly = im->region->ly;
  uy = im->region->uy;

  gthresh = (int)(SUSANMASK * BLUTSCALE * geothresh);
  /* should be 37 x 100 x 0.5 = 1850 */

  cofgx = im_alloc(im->height, im->width, im->region, int_v);
  cofgy = im_alloc(im->height, im->width, im->region, int_v);

  /* do susan principle pass */
  prnc_im = susan_prnc2(im, gthresh, blut);
  /* find true corners from principle image and also find centre */
  /* of gravtiy of usan areas at those max positions.*/
  corn_im = susan_cnrmnts(prnc_im, im, gthresh, blut, &cofgx, &cofgy);
  /* find corner position to sub-pix accuracy and its orientation */
  maxim = susan_locatcnr(corn_im, prnc_im, cofgx, cofgy);

  im_free(prnc_im);
  im_free(corn_im);
  im_free(cofgx);
  im_free(cofgy);
  return(maxim);
}
/* EOF */
```

Appendix E

Camera Models used in Tina

```
typedef struct camera  {

      Ts_id   ts_id;            /* Tina structure identifier */
      unsigned int type;
      unsigned int label;

      /** physical parameters **/
      float   f;       /* focal length */
      float   pixel;            /* notional pixel size */
      float   ax, ay;           /* x and y expansion factors (aspect ratio ) */
      float   cx, cy;           /* x and y image centre coordinates */
      int     width, height;    /* image height and width for which relevant */

      Transform3 *transf;       /* transformation from world to camera frame */

      void   *distort_params;        /** optical distortion **/
      void   *(*copy_dist_func)( );
      Vec2   (*distort_func)( );
      Vec2   (*correct_func)( );

      Mat3    cam_to_im;     /* projection from unit camera to image coordinates */
      Mat3    im_to_cam;     /* projection from image to unit camera coordinates */

 } Camera;
```

```
typedef struct parcam {

      Ts_id   ts_id; /* Tina structure identifier */
      unsigned int type;
      unsigned int label;

      float   f; /* notional focal length */
      float   I; /* interocular separation */
      float   pixel; /* notional pixel size */

      /* cameras and rectified counter parts */
      struct camera *cam1; /* original camera 1 */
      struct camera *rcam1; /* rectified camera 1 */
      struct camera *cam2; /* original camera 2 */
      struct camera *rcam2; /* rectified camera 2 */

      struct mat3 rect1; /* rectification matrix for camera 1 */
      struct mat3 derect1; /* derectification matrix for camera 1 */
      struct mat3 rect2; /* rectification matrix for camera 2 */
      struct mat3 derect2; /* derectification matrix for camera 2 */

      struct mat3 e; /* epipolar colineation matrix */

  } Parcam;
```

Appendix F

Quaternion Encoded 3D Rotations

In Cartesian form, a quaternion is usually represented as,

$$q = w + x\,\boldsymbol{i} + y\,\boldsymbol{j} + z\,\boldsymbol{k} \tag{F.1}$$

where w, x, y and z are all real and $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are the complex operators which obey,

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1 \qquad \text{and} \qquad \boldsymbol{ij} = \boldsymbol{k}, \boldsymbol{jk} = \boldsymbol{i}, \boldsymbol{ki} = \boldsymbol{j}, \boldsymbol{ji} = -\boldsymbol{k}, \boldsymbol{kj} = -\boldsymbol{i}, \boldsymbol{ik} = -\boldsymbol{j} \tag{F.2}$$

The quaternion representation for the rotation of a 3D coordinate frame is given by:

$$\mathbf{R}_q = \begin{bmatrix} q_0, & q_1, & q_3 \end{bmatrix} \qquad \text{from} \qquad \mathbf{Q} = \begin{bmatrix} q_0, & q_1, & q_2, & q_3 \end{bmatrix} \tag{F.3}$$

where,

$$q_0 = \cos\left(\theta/2\right), \qquad q_1 = r_0 \sin\left(\theta/2\right) \tag{F.4}$$

and

$$q_2 = r_1 \sin\left(\theta/2\right), \qquad q_3 = r_2 \sin\left(\theta/2\right) \tag{F.5}$$

r is the vector defining the axis of rotation and θ is the angle of rotation this axis.

A rotation matrix $\mathbf{R}$ is parameterized as:

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\left(q_1 q_2 + q_0 q_3\right) & 2\left(q_1 q_3 - q_0 q_2\right) \\ 2\left(q_1 q_2 - q_0 q_3\right) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2\left(q_2 q_3 + q_0 q_1\right) \\ 2\left(q_1 q_3 + q_0 q_2\right) & 2\left(q_2 q_3 - q_0 q_1\right) & q_0^2 - q_1^2 - q_2^2 - q_3^2 \end{bmatrix} \tag{F.6}$$

Bibliography

[AR84] J Altmann and H J P Reitböck.
A fast correlation method for scale- and translation-invariant pattern recognition.
IEEE Trans: Pattern Analysis and Machine Intelligence, 6(1):46–57, 1984.

[Ben88] J P Bentley.
Principle of Measurement Systems.
Longman Scientific and Technical, 2nd edition, 1988.

[But97] S Butterfield.
Reconstruction of Extended Environments from Image Sequences.
PhD thesis, School of Computer Studies, University of Leeds, 1997.

[Can86] J F Canny.
A computational approach to edge detection.
IEEE Trans: Pattern Analysis and Machine Intelligence, 8(6):679–698, Nov 1986.

[CC94a] H I Christensen and J L Crowley, editors.
Experimental Environments for Computer Vision and Image Processing.
World Scientific, 1994.

[CC94b] A F Clark and S W Chan.
Single-camera computational stereo using a rotating mirror.
In E Hancock, editor, *Proc.British Machine Vision Conference*, volume 2, pages 761–770, Uni.of York, UK, 13–16 Sept 1994. BMVA.

[Cla98] A F Clark.
Software tools for vision.
Technical report, Dept. of Electronics Systems Eng. University of Essex, for the EPSRC Summer School in Computer Vision at the Uni. of Surrey, revised. 1998.
Available from *CVOnline*.

[CLTS97] S Crossely, A J Lacey, N A Thacker, and N L Seed.
Robust stereo via temporal consistency.
In A F Clark, editor, *Proc.British Machine Vision Conference*, volume 2, pages 659–668, Uni.of Essex, UK, 8–10 Sept. 1997. BMVA.

[CP76] D Casasent and D Psaltis.
Position, rotation and scale invariant optical correlation.
Applied Optics, 15(7):1795–1799, 1976.

[CTS98] S Crossley, N A Thacker, and N L Seed.
Benchmarking of bootstrap temporal stereo using statistical and physical scene modeling.

In *Proc.British Machine Vision Conference*, volume 1, pages 346–355. BMVA, 1998.

[CVP97] *Proc.IEEE Conf.on Computer Vision and Pattern Recognition*, Puerto Rico, USA, 17–19 Jun 1997. IEEE Comp. Soc.

[Dav97] E R Davies.
Machine Vision: Theory, Algorithm, Practicalities.
Academic Press, 2nd edition, 1997.

[Der87] R Deriche.
Using Canny's criteria to derive a recursively implemented optimal edge detector.
Int. Journal of Computer Vision, 1(2):167–187, Apr 1987.

[Ekl94] J Eklundh, editor.
Proc.European Conf.on Computer Vision, LNCS vol.800/801, Stockholm, Sweden, 2–6 May 1994. Springer-Verlag.

[Fau92] O Faugeras.
What can be seen in three dimensional with an uncalibrated stereo rig.
In Sandini [San92], pages 563–578.

[Fau93] O D Faugeras.
Three-Dimensional Computer Vision - A Geometric Viewpoint.
MIT Press, 1993.

[FB81] M A Fischler and R C Bolles.
Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.
Assoc. of Computing Machinery, 24(6):381–395, 1981.

[FB87] M A Fischler and R C Bolles.
Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.
In Fischler and Firschein [FF87], chapter 7, pages 726–740.

[FF87] M A Fischler and O Firschein, editors.
Readings in Computer Vision: Issues, Problems, Principles and Paradigms.
Morgan-Kaufmann, 1987.

[FLM92] O Faugeras, Q T Luong, and S J Maybank.
Camera self-calibration: Theory and experiments.
In Sandini [San92], pages 321–334.

[För94] W Förstner.
A framework for low-level feature extraction.
In Eklundh [Ekl94], pages 383–395.

[FvDFH97] J D Foley, A van Dam, S K Feiner, and J F Hughes.
Computer Graphics: Principles and Practice.
Addison-Wesley, 2nd edition, 1997.

[GG93] A Goshtasby and W A Gruver.
Design of a single-lens stereo camera system.
Pattern Recognition, 26(6):923–936, 1993.

[Goa86] C Goad.
Fast 3D model-based vision, chapter From Pixels to Predicates, pages 371–374.
Ablex, Norwood, NJ, 1986.

[HÅ96] A Heyden and K Åström.
Euclidean reconstruction from constant intrinsic parameters.
In ICPR96 [ICP96], pages 339–343.

[HÅ97] A Heyden and K Åström.
Euclidean reconstruction from images sequences with varying and unknown focal length and principal point.
In CVPR97 [CVP97], pages 438–443.

[Ham66] W. H Hamilton.
Elements of Quaternions.
Longmans, Green and Co., 1866.

[Har84] R M Haralick.
Digital step edges from zero-crossing of second directional derivatives.
IEEE Trans: Pattern Analysis and Machine Intelligence, 6(1):58–68, Jan 1984.

[Har92] R Hartley.
Estimation of relative camera positions for uncalibrated cameras.
In Sandini [San92], pages 579–587.

[Har94] R Hartley.
Self-calibration from multiple views with a rotating camera.
In Eklundh [Ekl94], pages 471–478.

[Har95] R Hartley.
In defence of the 8-point algorithm.
In *Proc.IEEE Int.Conf.on Computer Vision*, pages 1064–1070, MIT Cambridge, USA, 20–23 Jun 1995. IEEE Comp. Soc.

[HB92] D Hogg and R Boyle, editors.
Proc.British Machine Vision Conference, Uni. of Leeds, UK, Sept 1992. BMVA.

[HC98] R Horaud and G Csurka.
Self-calibration and euclidean reconstruction using motions of a stereo rig.
In ICCV98 [ICC98], pages 96–103.

[HGC92] R Hartley, R Gupta, and T Chang.
Stereo from uncalibrated cameras.
In *Proc.IEEE Conf.on Computer Vision and Pattern Recognition*, Champaign, IL, USA, 15–18 Jun 1992. IEEE Comp. Soc.

[HMDB95] R Horaud, R Mohr, F Dornaika, and B Boufama.
The advantage of mounting a camera onto a robot arm.
In *Proc.Europe–China workshop on Geometrical Modelling and Invariants for Computer Vision*, pages 206–213, Xian, China, April 1995. Xidan University Press.

[HPBG94] M Hebert, J Ponce, T Boult, and A Gross, editors.
Proc.Int.NSF–ARPA Workshop on Object Representation in Computer Vision, LNCS vol.994, New York, NY, USA, 5–7 Dec 1994. IEEE, Springer-Verlag.

[HS88] C G Harris and M Stephens.
A combined corner and edge detector.
In *4th Alvey Vision Conference*, pages 147–151, Aug 1988.

[HZ00] R I Hartley and A Zisserman.
Multiple View Geometry in Computer Vision.
Cambridge University Press, 2000.

[ICC98] *Proc.Int.Conf.on Computer Vision*, Bombay, India, Jan 1998. IEEE Comp. Soc.

[ICP96] *Proc.Int.Conf.on Pattern Recognition*, Vienna, Austria, 25–30 Aug 1996. IEEE Comp. Soc.

[IH98] J Illingworth and A Hilton.
Looking to build a model world.
IEE Electronics and Communications Engineering, 10(3):103–113, Jun 1998.

[IYT92] H Ishiguro, M Yamamoto, and S Tsuji.
Omni-directional stereo.
IEEE Trans: Pattern Analysis and Machine Intelligence, 14(2):257–262, 1992.

[JKS95] R Jain, R Kasturi, and B Schunck.
Machine Vision.
McGraw-Hill, 1995.

[KR82] L Kitchen and A Rosenfeld.
Gray-level corner detection.
Pattern Recognition Letters, 1:95–102, 1982.

[KS97] S B Kang and R Szeliski.
3D scene data recovery using omnidirectional multibaseline stereo.
Int. Journal of Computer Vision, 25(2):167–183, 1997.

[KS99] K Kutulakos and A Shashua, editors.
Proc.IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes. part of CVPR99, IEEE Comp. Soc., 1999.

[KSK98] R Klette, K Schlüns, and A Korschan.
Computer Vision: Three-Dimensional Data from Images.
Springer-Verlag, 1998.

[KvG98] R Koch and L van Gool, editors.
Proc.European Workshop on 3D Structure from Multilple Images of Large-scale Environments, LNCS 1506. part of ECCV98, Springer-Verlag, 1998.

[LH81] H C Longuet-Higgins.
A computer algorithm for reconstructing a scene from two projections.
Nature, 293(10):133–135, Sept 1981.

[Lin99] M Lincoln.
Homogeneous Video Capture Interface.
Dept. of Electronics Systems Engineering, University of Essex, 1999.

[LTM94] R Lane, N A Thacker, and J E W Mayhew.
Stretch correlation as a real-time alternative to feature based matching algorithm.
Image and Vision Computing, 12(4):203–212, 1994.

[Mar82] D Marr.
Vision: A Computational Investigation into Human Representation and processing of Visual Information.
Freeman, 1982.

[MB92] D W Murray and P A Beardsley.
Range recovery using virtual multi-camera stereo.
In Hogg and Boyle [HB92], pages 29–38.

[MF92] S J Maybank and O D Faugeras.
A theory of self-calibration of a moving camera.
Int. Journal of Computer Vision, 8(2):123–151, 1992.

[MK98] D Morris and T Kanade.
A unified factorization algorithm for points, line segments and planes with uncertainty models.
In ICCV98 [ICC98], pages 696–702.

[Moh93] R Mohr, editor.
Projective Geometry and Computer Vision, chapter 2.4, pages 369–393.
World Scientific, 1993.

[MvGvDP93] T Moons, L van Gool, M van Diest, and E Pauwels.
Affine reconstruction from perspective image pairs obtained by a translating camera.
In Mundy et al. [MZF93], pages 293–316.

[MYI89] T Morita, Y Yasukawa, and Y Inamoto.
Measurement in three dimensions by motion stereo and spherical mapping.
In *Proc.IEEE Conf.on Computer Vision and Pattern Recognition*, pages 422–428, San Diego, CA, USA, 4–8 Jun 1989. IEEE Comp. Soc.

[MZ92] J L Mundy and A Zisserman, editors.
Geometric Invariance in Computer Vision.
MIT Press, 1992.

[MZ93] J L Mundy and A Zisserman.
Repeated structures: Image correspondence constraints and 3D structure recovery.
In Mundy et al. [MZF93], pages 89–106.

[MZF93] J L Mundy, A Zisserman, and D Forsyth, editors.
Proc.Joint Eu–US Workshop on Applications of Invariance in Computer Vision, LNCS vol.825, Ponta Delgada, Azores, Portugal, 9–14 Oct 1993. Springer-Verlag.

[NFJ93] T S Newman, P J Flynn, and A K Jain.
Model-based classification of quadric surfaces.
CVGIP: Image Understanding, 58(2):235–249, Sept. 1993.

[Nob88] J A Noble.
Finding corners.
Image and Vision Computing, 6(2):121–128, 1988.

[PBEP01] S Peleg, M Ben-Ezra, and Y Pritch.
Omnistereo: Panoramic stereo imaging.
IEEE Trans: Pattern Analysis and Machine Intelligence, 23(3):279–290, 2001.

[PFTV93] W. H Press, B. P Flannery, S. A Teukolsky, and W. T Vetterling.
Numerical Recipes in C: The Art of Scientific Computing.
Cambridge University Press, 2nd edition, 1993.

[PH95] T Pajdla and V Hlaváč.
Camera calibration and euclidean reconstruction from known translations.
presented at Computer Vision and Applied Geometry Workshop, Nordfjordeid, Norway, 1–7 Aug 1995.

[PH97] S Peleg and J Herman.
Panoramic mosaics by manifold projection.
In CVPR97 [CVP97], pages 338–343.

[PKVvG98] M Pollefeys, R Koch, M Vergauwen, and L van Gool.
Metric 3D surface reconstruction from uncalibrated image sequence.
In Koch and van Gool [KvG98], pages 139–154.

[PMF85] S B Pollard, J E W Mayhew, and J P Frisby.
PMF - A stereo correspondence algorithm using a disparity gradient limit.
Perception, 14:449–470, 1985.

[Pol00] M Pollefeys.
3D modelling from images - A tutorial.
Technical report, Katholieke Universiteit, Leuven, presented at ECCV00 in Dublin, Ireland, Jun 2000.

[Pra00] T Pratt.
From photo to 3D model.
IEE Review, 46(1):9–12, Jan 2000.

[PZ98] P Pritchett and A Zisserman.
Matching and reconstruction from widely separated views.
In Koch and van Gool [KvG98], pages 78–92.

[PZH96] J Ponce, A Zisserman, and M Hebert, editors.
Proc.Int.Workshop on Object Representation in Computer Vision, LNCS 1144, Cambridge, UK, 13–14 Apr 1996. part of ECCV96, Springer-Verlag.

[QK96] L Quan and T Kanade.
A factorization method for affine structure from line correspondence.
Computer Vision and Pattern Recognition, pages 803–808, 1996.

[Roc99] P Rockett.
The accuracy of sub-pixel localisation in the canny edge detector.
In T Pridmore and D Elliman, editors, *Proc.British Machine Vision Conference*, pages 392–401, Uni. of Nottingham, UK, 13–16, Sept. 1999. BMVA.

[RRT01] M J Rendas, S Rolfes, and A Tenas.
Autonomous mapping of natural and unstructured environments.
In *Proc.Int.Workshop on Underwater Robotics for Sea Exploitation and Environment Monitoring*, pages 22–33, Rio de Janeiro, Brasil, Oct. 2001.

[San92] G Sandini, editor.

Proc.European Conf.on Computer Vision, LNCS vol.588, Santa Marghertia Ligure, Italy, 19–22 May 1992. Springer-Verlag.

[SB97] S M Smith and J M Brady.
SUSAN - A new approach to low level image processing.
Int. Journal of Computer Vision, 23(1):45–78, 1997.

[SHB99] M Sonka, V Hlavac, and R Boyle.
Image Processing, Analysis and Machine Vision.
PWS Pub., 2nd edition, 1999.

[SK52] J. G. Semple and G. T. Kneebone.
Algebraic Projective Geometry.
Re-pub. under Oxford Classics. Oxford University Press, reprinted 1979 edition, 1952.

[Smi97] S Smith.
Review of optic flow, motion segmentation, edge finding and corner finding.
Technical Report TR97SMS1, Oxford Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB), Dept. of Clinical Neurology, Oxford University, 1997.
for *CVOnline*.

[Spa96] G Sparr.
Simultaneous reconstruction of scene structure and camera locations from uncalibrated image sequences.
In ICPR96 [ICP96], pages 328–333.

[ST98] R Szeliski and P H S Torr.
Geometrically constrained structure from motion: Points on planes.
In Koch and van Gool [KvG98], pages 171–186.

[Sze96] R Szeliski.
Video mosaics for virtual environments.
IEEE Computer Graphics and Applications, 16(2):22–30, 1996.

[TC92] N A Thacker and P Courtney.
Statistical analysis of stereo matching algorithm.
In Hogg and Boyle [HB92], pages 316–326.

[TC96] N A Thacker and T F Cootes.
Vision through optimization.
Tutorial written for BMVA, Uni. of Manchester, 1996.
available at *CVOnline*.

[TFB80] J M Tenenbaum, M A Fischler, and H G Barrow.
Scene modeling: A structural basis for image description.
In A. Rosenfeld, editor, *Image Modeling*, pages 371–389. Academic Press, 1980.

[TJNU97] G A Thomas, J Jin, T Niblett, and C Urquhart.
A versatile camera position measurement for virtual reality TV production.
In *Proc.International Broadcasting Convention*, pages 284–289, Amsterdam, 12–16 Sept. 1997. IEE.

[TM91] N A Thacker and J E W Mayhew.
Optimal combination of stereo calibration from arbitary stereo images.

Image and Vision Computing, 9(1):27–32, Feb 1991.

[TM93] P H S. Torr and D W Murray.
Outlier detection and motion segmentation.
In P. S. Schenker, editor, *Proc.Sensor Fusion VI*, pages 432–443. SPIE vol.2059, 1993.

[TM97] P H S Torr and D W Murray.
The development and comparison of robust methods for estimating the fundamental matrix.
Int. Journal of Computer Vision, 24(3):271–300, 1997.

[Tri87] H P Trivedi.
Estimation of stereo and motion parameters using a variational principle.
Int. Journal of Computer Vision, 5(2):181–183, May 1987.

[Tri97a] B Triggs.
Autocalibration and the absolute quadric.
In CVPR97 [CVP97], pages 609–614.

[Tri97b] B Triggs.
Linear projective reconstruction from matching tensors.
Image and Vision Computing, 15(8):617–625, Jun 1997.

[Tsa87] R Y Tsai.
A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf cameras and lens.
IEEE Trans: Robotics and Automation, 3(4):323–344, Aug 1987.

[Wil94] R G Willson.
Modeling and calibration of automated zoom lenses.
In *Proc.of the SPIE: Videometrics III*, volume 2350, pages 170–186, Boston MA, Oct 1994.

[YK90] Y Yagi and S Kawato.
Panorama scene analysis with conic projection.
In *Proc.IEEE Int.Workshop on Intelligent Robotic Systems*, pages 181–187. IEEE, 1990.

[ZDFL95] 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.
Artificial Intelligence, 78:87–119, Oct 1995.

[ZFD97] Z Zhang, O D Faugeras, and R Deriche.
An effective technique for calibrating a binocular stereo through projective reconstruction using both a calibration object and the environment.
Journal of Computer Vision Research (MIT Press), 1(1):58–68, 1997.

[Zha98] Z Zhang.
Determining the epipolar geometry and its uncertainty - A review.
Int. Journal of Computer Vision, 27(2):161–195, Mar 1998.