UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

School of Electronics and Computer Sciences

Camera Self-Calibration for Augmented Reality

by

Junaidi Abdullah

Thesis for the degree of Doctor of Philosophy

September 2005

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE, AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Junaidi Abdullah

One of the characteristics of a good Augmented Reality (AR) system is to be able to register virtual objects correctly onto the specified location in the real world. A displacement from this specified location is called registration error. This error can be caused by several factors, such as system delay, optical distortion and poor camera calibration. The problem of poor camera calibration in AR has always been overcome by employing careful camera calibration steps with the use of a specific known object. However this task is time consuming and mostly performed offline. This dissertation aims to develop an AR system known as a SCAR (Self-Calibration for Augmented Reality) system, which incorporates a self-calibration of a camera so that the system updates the camera intrinsic parameters whenever they change. The SCAR system incorporates an algebraic approach to self-calibration where it can solve camera parameters based on only three views and requires only the fundamental matrices as the inputs. The solution proposed here can be used for any AR system that uses visual-based tracking. Several pre-calibration stages including feature detection, point correspondence matching, and fundamental matrix estimation are developed. The problem of inaccuracy with general corner detector has been identified and a new algorithm, which combines Harris and SUSAN corner detectors, has been suggested. This hybrid detector increases the corner detection accuracy and reduces

localisation errors of up to several pixels. A novel point correspondence matching has been developed, which is based on motion vector and simple statistic calculation. The matching process is efficient and capable of removing outliers from corner detection as well as maintaining a good number of correct matches even in the event of occlusion. Lens distortion is a common problem in visual-based tracking. This problem is dealt with by solving general epipolar constraints in order to simultaneously solve for the distortion parameters and fundamental matrix using MAPSAC algorithm. Finally the SCAR system is compared with the ARToolKit calibration procedure and has proven to produce reliable results with better flexibility. The theoretical aspects of critical motion, which may exist between pairs of views, are also discussed. A new measure of the criticalness of the motion for the case of parallel camera axis followed by rotation along the principal axis is also presented.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURES

Operators

$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$.
$\mathbf{A}^{T}$	Transpose of matrix $\mathbf{A}$.
$\det(\mathbf{A})$	Determinant of matrix A.
$\mathrm{svd}(\mathbf{A})$	Singular value decomposition of matrix $\mathbf{A}$.
$trace(\mathbf{A})$	Sum of diagonal element of A.

Variables

b	Degree of slant of the co-ordinate axis u in the camera image plane.
$d(u,v)$	Distance between an estimated corner and reference corner in pixel.
$\mathbf{e},\mathbf{e}'$	Epipoles in the first and second image respectively.
$\mathbf{E}$	Essential matrix.
$\mathbf{F}$	Fundamental matrix.
f	Focal length.
h	Distance of adjacent pixel in vertical direction.
$\mathbf{H}$	The matrix used by Harris detector to determine corner response function.
$I(u,v)$	Pixel brightness value at position (u,v).
$\mathbf{K}$	Camera calibration matrix.

$k_1, k_2, k_3, \ldots$	Coefficients of radial distortion.
$\mathbf{M}$	4x4 rotation and translation transformation matrix.
m	A fixed value set to 0.04 in Harris corner response function.
$n(\vec{r}_0)$	Total number of pixels in the USAN.
$n_{\max}$	Total number of pixel inside the SUSAN mask.
$\mathbf{O}_w$	Origin of world Euclidean co-ordinate system.
$\mathbf{O}_c$	Origin of camera Euclidean co-ordinate system.
$\mathbf{P}$	3x4 projective matrix
$p_1, p_2, p_3, \ldots$	Coefficients of de-centring distortion.
$\mathbf{q} = (q_0, q_x, q_y, q_z)$	Quaternion
$\mathbf{R}$	3x3 rotation matrix from world co-ordinate axis orientation to camera co-ordinate axis orientation.
$\mathbf{R}_f$	3x3 rotation matrix from first camera co-ordinate axis orientation to second camera co-ordinate axis orientation.
r	Radial distance of a point from principal point.
$\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$	Row vectors of rotation matrix $\mathbf{R}$.
$\vec{r}_0$	Nucleus pixel in SUSAN mask.
$\vec{r}$	Pixels other than nucleus in SUSAN mask.
S_{ref}, S_{est}	Set of reference corner and set of estimated corner respectively.
s	Scale factor.
$\mathbf{t}$	3x1 translation vector from centre of world co-ordinate axis to centre of camera co-ordinate axis.

	ordinate axis to centre of camera co-ordinate axis.
$\mathbf{t}_J$	3x1 translation vector from centre of first camera co-ordinate axis to centre of second camera co-ordinate axis.
t	Threshold for pixel brightness difference in SUSAN mask between $I(\vec{r}_0)$ and $I(\vec{r})$.
u, v	2D image affine co-ordinate system.
$(\hat{u}_c, \hat{v}_c)$	Ideal point in 2D camera Euclidean co-ordinate system.
$\tilde{\mathbf{U}}_a = (u_a, v_a, 1)^T$	A point in 2D camera affine co-ordinate system expressed in homogeneous co-ordinate.
$\mathbf{U}_a = (u_a, v_a)^T$	A point in 2D camera affine co-ordinate system.
$\mathbf{U}_{a0} = (u_0, v_0)^T$	Principal point. A point in 2D camera affine co-ordinate system represents the centre co-ordinate of an image plane.
$\mathbf{U}_c = (u_c, v_c)^T$	A point in 2D camera Euclidean co-ordinate system.
$\tilde{\mathbf{U}}_a, \tilde{\mathbf{U}}'_a$	Normalised points in 2D camera affine co-ordinate system for left and right image respectively.
w	Distance between adjacent pixel in horizontal direction.
$\mathbf{X}_c = (x_c, y_c, z_c)^T$	A point in 3D camera Euclidean co-ordinate system.
$\tilde{\mathbf{X}}_c = (x_c, y_c, z_c, 1)^T$	A point in 3D camera affine co-ordinate system expressed in homogeneous co-ordinate.
$\mathbf{X}_w = (x_w, y_w, z_w)^T$	A point in 3D world Euclidean co-ordinate system.
X_w, Y_w, Z_w	3D World Euclidean co-ordinate system.
X_c, Y_c, Z_c	3D Camera Euclidean co-ordinate system.

α_u	Horizontal scale factor. Ratio of focal length to width of a pixel.
α_v	Vertical scale factor. Ratio of focal length to height of a pixel.
δ_u, δ_v	Distortion correction to $(\hat{u}_c, \hat{v}_c)$.
σ	Standard deviation.
Ω	Harris corner response function.

ACKNOWLEDGMENTS

I am indebted to my supervisor, Dr. Kirk Martinez, for his supervision and guidance in the course of my research.

I am also very grateful to Majlis Amanah Rakyat and Multimedia University for their financial aid.

Last but not least, special thanks and kisses go to my wife Wan Noorshahida for her love and continuous support throughout my whole university career.

Chapter 1

INTRODUCTION

1.1 Augmented Reality

1.1.1 Definitions

In the past few years, there has been considerable interest in mixing live video from a camera with computer-generated graphical objects that are registered in a user's three-dimensional environment; this process is called Augmented Reality (AR) [Azuma 1997]. This is due to emerging developments in Virtual Reality (VR) and wearable computing. People in virtual reality are extending their research on fully virtual immersive system into research that combines virtual objects in real scenes, i.e. AR, or combines real objects in virtual scenes, i.e. Augmented Virtuality (AV). The increasing development of wearable devices, from monitor-based displays to head-mounted displays (HMDs), is having a significant impact on AR research. HMDs make it possible for the AR system to function outdoors as well as in the laboratory environment.

Augmented Reality is a combination of the real scene viewed by the user and a 3D virtual object generated by the computer, which augments the user's view of the real world. According to Azuma [1997], an AR system is a

system that has the following three characteristics: one, it combines real and virtual; two, it is interactive in real time; and three, it is registered in 3D.

Therefore, overlaying 2D virtual objects onto the real world cannot be considered AR. Films like Jurassic Park also cannot be regarded as AR, because they are not in an interactive medium. Figure 1.1 shows an example of AR where a man is interactively realising a car body in Spacedesign using 3D devices and AR/VR HMD.

Virtual Reality (VR) is an area that is closely related to AR. AR differs from VR due to the fact that it brings the computer into the 'world' of the user (compositing real and virtual), rather than immersing the user in the world of the computer (virtual only). A Milgram's Reality-Virtuality Continuum [Milgram and Kishino 1994] shown in Figure 1.2 illustrates how real and virtual worlds are combined in various proportions. AR has the real world as the background plus some virtual objects, whereas AV is vice versa. Virtual Environment represents VR and Real Environment represents the real world.

1.1.2 Applications

Some of the target application domains for AR include computer-aided surgery, repair and maintenance of complex engines, facilities modification, education, the games and entertainment industry and many more. For example, in medical applications AR images of MRI-derived models could be overlaid on top of a surgeon's view of a patient during surgery to help identify malignant tissue to be removed or sensitive healthy areas to be avoided [Tuceryan et al. 1995]. Lorensen et al. [1993] and Figl et al. [2002] also used an AR system for surgical planning applications.

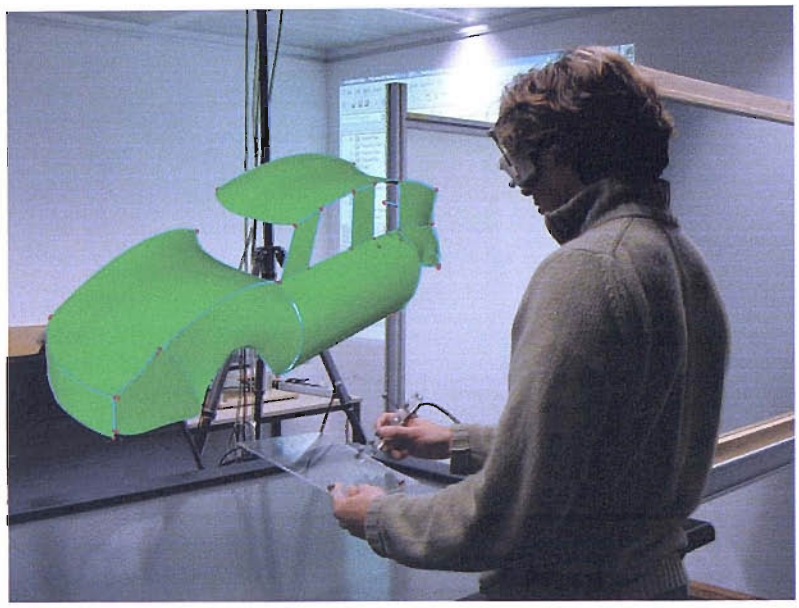


Figure 1.1: An example of Augmented Reality [Fiorentino et al. 02].

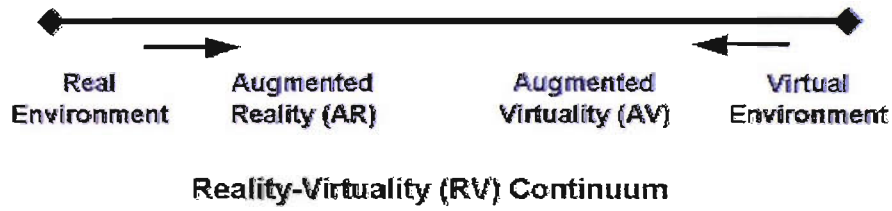


Figure 1.2: Milgram's Reality-Virtuality Continuum

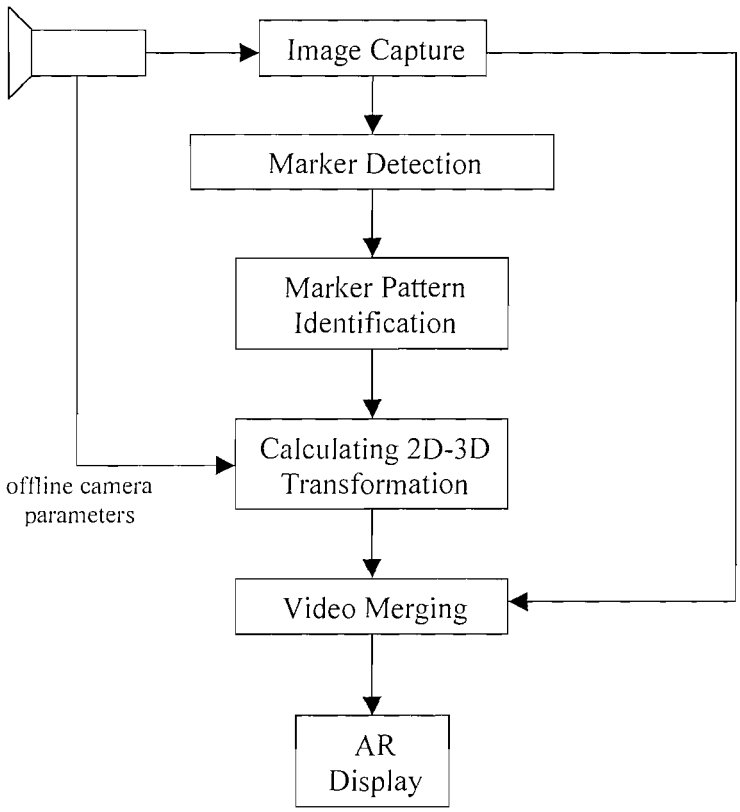


Figure 1.3: A typical AR system based on the ARToolKit architecture.

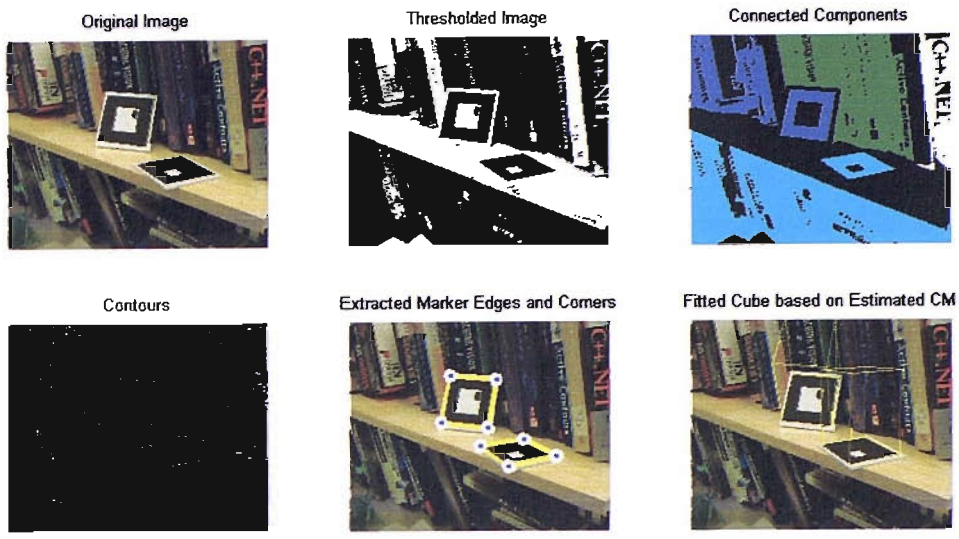


Figure 1.4: Four steps involved in the **marker detection stage** in the ARToolKit, which consist of thresholding the captured image; finding connected components and contours; extracting marker edges and corners.

Feiner et al. [1993] used AR in a laser printer maintenance task where the user is guided in the steps required to open the printer and replace various parts. Wellner [1993] developed an AR system in an office environment by overlaying a virtual desktop on a physical desk. Drascic et al. [1993] and Milgram et al. [1993] performed Telerobotics tasks by using an AR system with computer-generated stereo graphics.

More recent applications include using AR in a classroom as a visual aid for students to understand Geography [Shelton and Hedley 2002], using AR for an exhibition in a museum [Grafe et al. 2002], and an interactive AR theatre that gives users a novel theatre experience through the exciting features of human-to-human social and physical interaction [Cheok et al. 2002].

1.1.3 Augmented Reality System

Figure 1.3 shows a typical AR system. It illustrates the system that is being used in ARToolKit, software developed by the Human Interface Technology Lab in Washington (this can be downloaded from [ARToolKit]). ARToolKit was first developed in late 1998 for a collaborative AR project at the University of Washington's Human Interface Technology Laboratory. Since that time, it has grown to be used at dozens of research and academic institutions by hundreds of developers in various AR applications.

The system consists of a live camera taking real world scenes as the input into the system. The system then tries to detect markers from the input stream (refer Appendix A for examples of the marker patterns used in ARToolKit). After a marker has been detected, the pattern inside the marker is identified so that the system knows which marker corresponds to which 3D virtual object. Normally, one pattern represents only one virtual object. The orientation of the marker is also determined during the marker pattern identification stage. Finally, after the camera transformation (from 3D co-ordinate to 2D co-ordinate and vice versa) has been calculated, this 3D object is rendered on the pattern and shown on the AR display (refer Appendix A). The system will then continually repeat the process of detecting markers, identifying marker patterns, calculating camera transformations and merging video, so that it can track any changes in the position and orientation of the pattern. If changes occur, the 3D virtual object is rendered according to the new position and orientation of the pattern.

In order to render the virtual object according to the position and orientation of the pattern, the camera has to be calibrated. This is usually done before the system runs. The purpose of camera calibration is to provide the system with the intrinsic camera parameters, which include focal

length, aspect ratios and the centre of the image plane. These parameters are important for the system to calculate the 3D to 2D transformation and vice versa. A more detailed explanation of camera calibration will be given in Chapter 2.

Once the camera has been calibrated, the AR system is said to be able to do 3D tracking. Tracking in AR terms means observing movement by knowing the position and orientation of the object or pattern being tracked in space in real time. There are two ways of tracking in an AR system:

a) Vision-based tracking

This type of tracking depends on camera vision. An object will be tracked if it is positioned within the camera view. The advantage of vision-based tracking is that it utilises the very same image or pattern on which virtual objects are overlaid. Therefore, nearly perfect registration can be achieved under certain conditions [Uenohara and Kanade 1995; Mellor 1995b]. Vision-based tracking outperforms other kinds of tracking in terms of the accuracy of registration. ARToolKit uses this form of tracking to track markers and patterns.

b) Magnetic tracking

Most tracking systems used today in fully immersive VR systems have been magnetic. The disadvantage of using magnetic trackers is that they produce large amounts of error and jitter. An uncalibrated system may exhibit position errors of 10 cm or more, particularly in the presence of magnetic field disturbances such as metal and electric equipment. Carefully calibrating a magnetic system can reduce position errors to within 2 cm [Livingston and State 1995]. Despite their lack of accuracy, magnetic trackers are popular because they are robust and place minimal constraints on user motion.

1.1.4 AR Display Technology

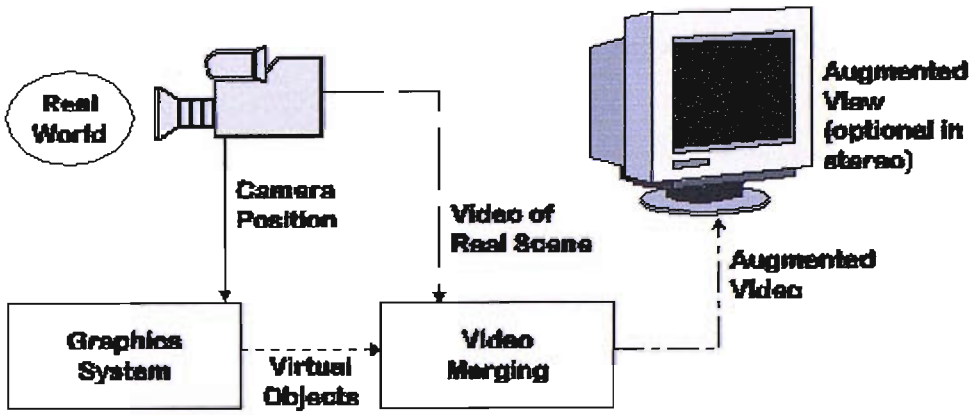


Figure 1.5: Monitor based AR display [Vallino 2002].

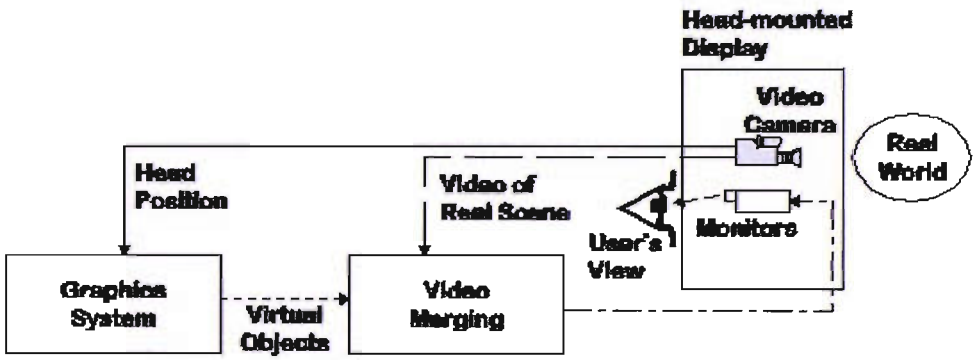


Figure 1.6: Video see-through AR display [Vallino 2002].

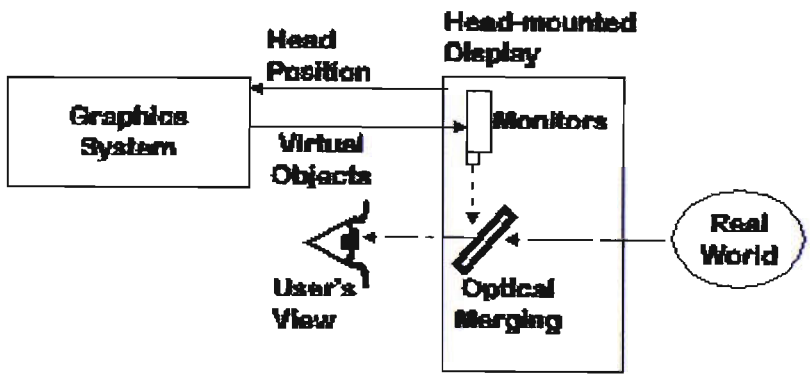


Figure 1.7: Optical see-through AR display [Vallino 2002].

There are currently three types of display technology used in AR systems, namely monitor-based AR display (Figure 1.5), video see-through AR display (Figure 1.6) and optical see-through AR display (Figure 1.7).

In general, a monitor-based AR display is employed in a situation where the sense of presence is not important. The display is visually isolated from the system and typically not aligned with the camera. In situations where the user wants to feel more sense of presence, Head-Mounted Displays (HMDs) are often used. Figures 1.6 and 1.7 illustrate two types of HMD, video see-through and optical see-through.

Figures 1.5 and 1.6 illustrate that the architecture of video see-through is very similar to that of the monitor-based display except that in video see-through the camera is aligned with the monitor display and is placed on the HMD. This enables the user to see an augmented worldview that is immediately in front of his eyes. However, it does not allow any direct view of the real world because the real world view comes from the monitor.

Optical see-through HMD allows the user to observe the real world, with virtual objects superimposed by optical or video technologies. It operates by placing an optical combiner in front of the user's eyes, which is partially transmissive and partially reflective. An optical approach HMD has the following advantages over a video approach HMD [Azuma 1997]:

a) *Simplicity:* Optical blending is simpler and cheaper than video blending.
b) *Resolution:* Optical see-through shows the virtual images at the resolution of the display device, but the user's view of the real world is not degraded.
c) *Safety:* In video see-through, if the power is cut off, the user is effectively blind, which may be dangerous in certain applications.

d) *No eye offset:* In most configurations of video see-through, the cameras are not located exactly where the user's eyes are, creating an offset between cameras and the real eyes.

In contrast, a video approach HMD has the following advantages over an optical approach HMD [Azuma 1997]:

a) *Flexibility in composition strategies:* In optical see-through, the virtual objects do not completely obscure the real world objects. This makes the user observe the virtual object as if a 'ghost object' were overlaid on the real scene. Unless a new technology of optical combiner can be created to overcome this, people will prefer the video see-through display for their applications.
b) *Real and virtual view delays can be matched:* In video see-through, it is possible to delay the video of the real world to match the delay from the virtual image stream. This can reduce the registration error that comes from the system delay.

1.2 State of the Art in Calibration for AR

1.2.1 Offline Plane-Based Camera Calibration

One of the calibration techniques in AR that is most commonly used is offline plane-based calibration. Figure 1.8 shows an example of a plane used for calibrating a camera. The motivations for considering planes for calibrating cameras are mainly twofold [Sturm and Maybank 1999]. First, planar calibration patterns are cheap and easy to produce. Second, planar surface patches are probably the most important two-dimensional features. If their metric structure is known, they already carry enough information to determine a camera's position up to only two solutions in general [Holt and Netravali 1991]. This method yields a very accurate determination of the

camera parameters, provided the calibration pattern is carefully set [Faugeras et al. 1992].

The disadvantage of the method is that it is not possible to calibrate on-line when the camera is already involved in a visual task. This can happen when the intrinsic parameters are changed either intentionally, for example during adjustment of focal length, or unintentionally, for example due to the effect of thermal variation. ARToolKit software uses this kind of calibration technique to calibrate the camera [ARToolKit].

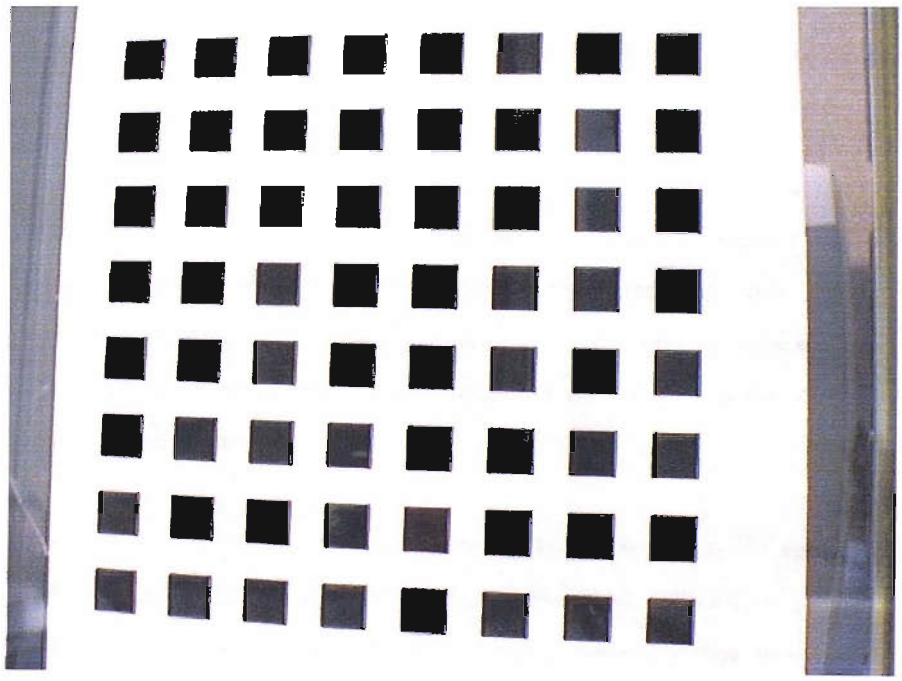


Figure 1.8: Example of calibration pattern.

1.2.2 Magnetic Tracker Calibration

Azuma and Bishop [1994] demonstrated accurate static registration across a wide variety of viewing angles and position by using an optoelectronic tracker for optical see-through HMD. Since optical see-through does not need a camera, three calibration steps were created to directly measure the viewing parameters, using simple tasks that rely on geometric constraints. Dynamic errors are reduced by predicting future head locations using inertial sensors mounted on the HMD. Even though this method achieved high accuracy, they require many technical steps and extra hardware devices, making the system demanding of time and concentration for the user.

1.2.3 Offline Hybrid Calibration (Camera and Magnetic Tracker)

Bajura [1992] proposed calibration for video see-through systems based on tracking known features in the working environment. Two types of transformation, consisting of transducer transformation and camera transformation (for position and orientation), are used to calibrate the system. However, there is no information on how the cameras' intrinsic parameters are found.

In Tuceryan et al. [1995], the calibration method developed for monitor-based augmented reality systems at ECRC (GRASP System) relies on a specific pattern. The dependency on a specific pattern makes this system only suitable for use indoors or inside a laboratory. The use of a magnetic tracker makes the whole calibration process lengthy and expensive.

In ARGOS (Augmented Reality through Graphic Overlays on Stereo-video), semi-automatic calibration is used to calibrate stereoscopic cameras for an Amiga-based AR system [ARGOS]. This calibration assumes that the

user has prior knowledge about the focal length of the camera. The inclusion of manual steps requires a lot of time and effort to accurately measure the parameters needed for the camera.

1.2.4 Other Calibration

Fuhrmann et al. [2000] propose an HMD calibration, which consists of displaying points in the corners of each HMD display and aligning these points with the hotspot of a stylus. The measured data gives all the necessary calibration information – eye position, view-plane distance, and aspect ratio.

Grasset et al. [2001] proposed an augmented reality system dedicated to the kind of collaborative applications where users meet around a table. The calibration technique used is for optical-based HMDs and is very simple and intuitive, at the cost of a loss in accuracy. The achieved accuracy is only sufficient for non-critical applications like architectural design, gaming and planning simulation.

Yao and Calway [2002] proposed a method of estimating 3D camera motion based on sparse feature tracking and recursive structure from a motion algorithm developed by Azarbayejani and Pentland [1995]. The method is highly dependent on the ability of the feature tracker to track the points making it unreliable when the points being tracked are lost due to being obscured by other objects.

1.2.5 No Calibration

There is an approach in AR, which avoids the need for any calibration [Kutulakos and Vallino 1996; Kutulakos and Vallino 1998]. Kutukalos

represents virtual objects in a non-Euclidean affine frame of reference that allows rendering without knowledge of camera parameters. However it should be noted that this approach might not recover all the information required to perform all potential AR tasks. For instance, this approach does not recover true depth information, which is useful when compositing the real and the virtual.

1.3 Motivations

The most likely problem faced in AR applications is achieving accurate registration, in which the virtual object must be properly aligned with the real world [Azuma 1993]. There are two types of error source that cause registration problems: static and dynamic [Holloway 1995]. Static errors are ones that cause registration errors when both the user's viewpoint and the object in the environment remain still. Dynamic errors are ones that cause registration errors when either the viewpoint or the objects begin to move.

One way to minimise registration errors is by having good camera calibration. Good calibration will ensure that the correct 2D camera to 3D world relationship is established. This will guarantee the accurate registration of the virtual object and hence produce less registration error. In order to achieve this, certain criteria must be met. Tsai [1987] stated that the calibration procedure should be an autonomous process, meeting certain accuracy requirements, reasonably efficient and versatile. Therefore, any work to develop good calibration techniques should have at least the aforementioned criteria in mind.

Serious research on camera calibration for AR only began less than a decade ago (refer section 1.2 for a complete review of camera calibration in AR). Before then, AR researchers concentrated on developing AR systems as a whole, with a minor focus on camera calibration. Therefore, most AR

systems employ the traditional way of calibrating a camera for AR by using calibration objects (refer Chapter 2 for more details on camera calibration). This is normally done offline and may **require a lot of time to complete**. This will produce more problems when the focal length of the camera changes either intentionally or unintentionally.

Additionally, the significant development of AR in various applications such as games requires the calibration process to be fast, simple and flexible. This cannot be achieved with offline calibration where the process **requires a specific known pattern, a special setup and certain movements**. If offline calibration were used, the excitement of children playing games would diminish, as they would have to keep stopping during their play merely to recalibrate the camera.

To solve these problems in AR, we propose a novel AR system that has a more flexible way of calibrating a camera without a known pattern, which can be done online. To have an AR system with a more flexible approach is worthwhile because of the fast emerging development of wearable computing such as HMDs. People will no longer want to bring along a pattern whenever they want to calibrate a camera. The system should be able to calibrate the camera any time and anywhere, while performing the AR task.

1.4 Aims and Scope

The aim of the research is **to develop a method of camera self-calibration based on a moving camera for an augmented reality system**. This means that the AR system will be able to update the intrinsic parameters of a camera while performing its AR task. An AR system that integrates camera self-calibration into the system has the advantage of being

flexible while avoiding the cumbersome steps that need to be done if plane-based calibration is used.

Since the aim is to integrate camera self-calibration into an AR system as a whole, the research will not only focus on the calibration stage, **but also on the pre-calibration stages**. This is because the success of self-calibration will also depend on the pre-calibration stages, which supply the input for the calibration stage. The main pre-calibration stages include the feature detection stage, the feature correspondence matching stage and the fundamental matrix estimation stage. The proposed pre-calibration stages involved and how they are integrated into AR system are illustrated in Figure 1.9. The calibration stage is shown in green and the pre-calibration stages are shown in yellow. The stages involved are described as follows:

Feature Detection

In the feature detection stage, we choose to focus on corner detection. Although other features such as line or profile may well be used in computer vision to estimate the fundamental matrix [Mendonca 2002], we believe that corners are the most widely found features in many situations. Our focus has been on increasing the accuracy of the detection. This is because the fundamental matrix estimation stage is quite sensitive to the accuracy of corners detected. Therefore, rather than developing a totally new algorithm for corner detection we focus on finding a new algorithm to increase the accuracy of detection using existing corner detectors.

Feature Correspondence Matching

In the feature correspondence matching stage, we focus on ways to discard false corners and false matches. No further focus on the accuracy of the location of the matches is made, since this is dealt with in the feature detection stage. The simplicity of the algorithm was kept in mind when

creating the algorithm, as speed is one of our concerns in real-time AR applications.

Fundamental Matrix Estimation

In this stage we use the MAPSAC algorithm proposed by Torr [2002].

Calibration Matrix Estimation

In the calibration stage, we propose the use of the algebraic approach algorithm based on three views for finding the intrinsic parameters of the camera being used for augmented reality. The algebraic constraints are chosen because of the simplicity of derivation and implementation compared with those based on geometrical constraints [Luong and Faugeras 97], in which calibration parameters are estimated by minimising the geometric distance between features and epipolar lines (a more complicated criterion). Its reduced sensitivity to initialisation parameters for non-linear optimisation makes it more reliable than other techniques that are highly dependent on good initialisation parameters in order to get correct results.

As for AR display, for the present this research will focus on the monitor-based augmented reality display as a starting point, before investigating other types of AR display such as video see-through and optical see-through. This means that the computer-generated graphics are combined with a live video signal to produce an enhanced view of a real scene, which is then displayed on a standard video monitor. This is because the technology of HMD for AR is still in its infancy.

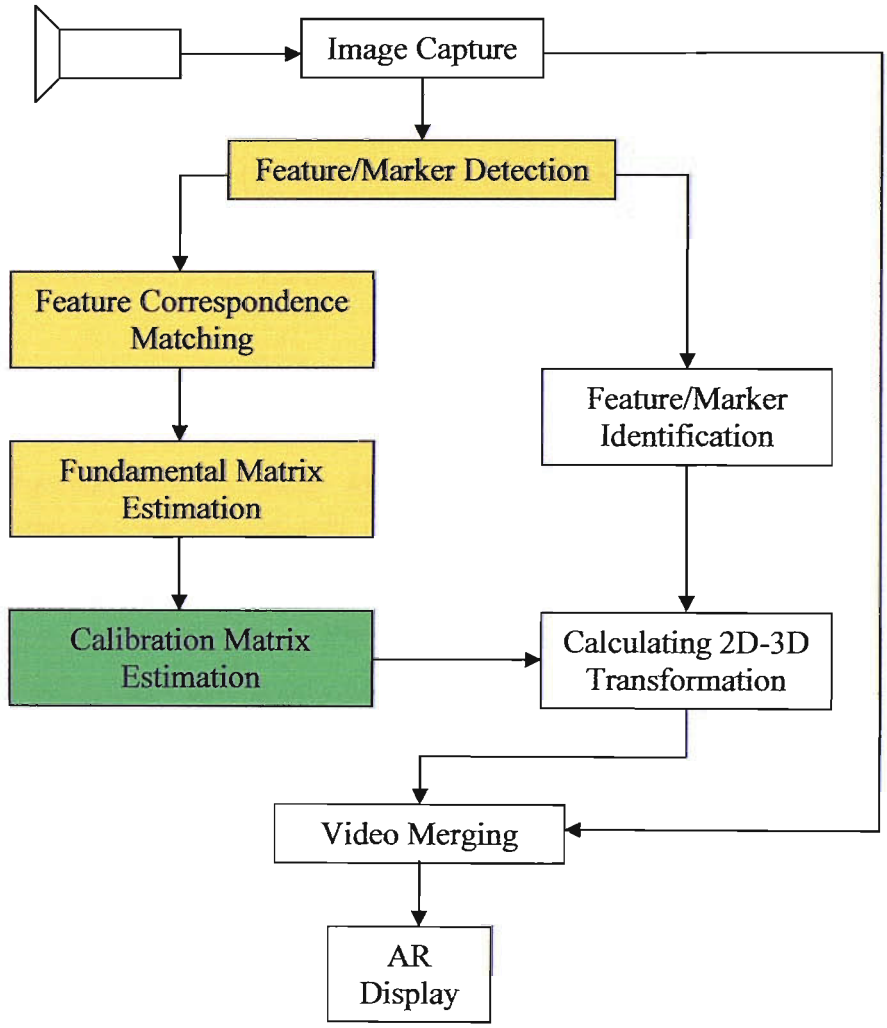


Figure 1.9: New stages included in AR system to accommodate self-calibration in AR

In addition, this research also aims to focus on improving the calibration results, particularly when dealing with camera distortion. This is especially important if wide-angle lenses, which have severe distortion, are used in AR applications. Severe distortion cannot simply be neglected, as it will result in the virtual object being overlaid incorrectly. We focus on ways to integrate distortion parameter estimation with fundamental matrix estimation stage, as done by Zhang [1996]. This makes the system more automatic, rather than having to estimate distortion parameters offline, which involves user intervention as in ARToolKit.

Finally, this research aims to look at the critical motion issue in camera self-calibration for AR. Sturm [2002] pointed out that there exist several motion sequences that are critical when solving camera parameters using self-calibration. This is important, as self-calibration based on a moving camera will produce false results if critical camera motion sequences are used as input into the AR system. The research aims to detect these critical motion sequences and consequently select only the particular frames to be used as an input into self-calibration stages. This is to ensure that the results gained are as close as possible to the correct values.

1.5 Outline and Contributions

This thesis is organised as follows:

- Chapter 2 presents the theory of camera self-calibration. This includes details on camera model, distortion model and epipolar geometry from which is derived the basis of camera self-calibration.

- Chapter 3 focuses on corner detection. A literature review of the available techniques for edge and corners detection is presented. Two state of the art corner detectors are chosen for their suitability for incorporation in an AR system and the algorithms of the two detectors

are discussed. A new hybrid Harris-Susan corner detection method is introduced to increase the accuracy of detection. Comparisons with other methods are presented.

- Chapter 4 touches on point correspondence matching. It discusses the issue of false matching and how it is to be overcome. We then present a novel way of discarding false matches that is simpler and more efficient. Comparison between the proposed method and other methods is also shown.

- Chapter 5 proposes the algebraic approach to camera self- calibration based on three views for AR. It then shows how the method can be extended to a case that has more than three views. The method derivation is explained and the algorithm of the proposed method is presented. It then touches on the issue of critical motion and suggests a new way to detect critical motion resulted from two cameras with parallel principal axis.

- Chapter 6 presents the experiments based on the algorithm described in Chapter 2, 3, 4 and 5 combined together to form the whole system. This includes discussion of the comparison between the proposed method and the calibration routine in the ARToolKit. The effect of camera distortion on the proposed method is also presented. Some related issues on iteration are also discussed.

- Chapter 7 presents a summary of the work carried out and conclusions, and points to future research directions in this area.

Original contributions of this thesis are presented as follows:

- A novel approach to **automated camera calibration** in Augmented Reality systems has been developed. In particular this research has explored the use of self-calibration for monitor-based AR displays [Abdullah and Martinez 02].

- A new **Harris-SUSAN hybrid corner detector** has been developed to reduce corner localisation error and increase detection accuracy.

- A new **point correspondence matching technique** has been developed based on motion vector and mode calculation to find matching features in AR applications. This simple technique outperforms other matching methods in terms of simplicity and efficiency.

- An improved **pre self-calibration stage through the integration of distortion parameters estimation and fundamental matrix estimation** has been developed. This is based on the use of MAPSAC algorithm to minimise the epipolar geometry constraint in order to simultaneously estimate the fundamental matrix and distortion parameters.

- We present theoretically **a new way of detecting critical motion sequence** that result from two views with parallel principal axes and how to overcome it.

A paper relating to this study has been published as "Camera Self-Calibration for the ARToolKit", in *The First IEEE International Augmented Reality ToolKit Workshop*, Darmstadt, Germany, September 2002.

Chapter 2

THEORY OF CAMERA SELF-CALIBRATION

2.1 Introduction

This chapter presents some of the theories that are necessary to understand the work carried out in the remainder of this thesis. The author has no intention of presenting a comprehensive survey of the theories since they can be found in so many books on projective geometry [Semple and Kneebone 1998; Springer 1964; Coxeter 1969; Coxeter 1974] and computer vision [Beardsley 1992; Mundy and Zissermann 1992; Faugeras 1993; Hartley and Zisserman 2000]. A reader who is already proficient in these two subjects may want to skip this chapter and continue with Chapter 3.

The first part of this chapter describes the mathematical model of a pinhole camera. The model explains the intrinsic and extrinsic parameters of a pinhole camera and how they are derived; an understanding of this is important for familiarity with how a camera is normally calibrated. It then discusses the different types of distortion that exist in a normal camera lens and how they are mathematically modelled. This is the basis for the reader to understand further Chapter 5, which discusses how distortion correction is inserted as an integral part of the self-calibration.

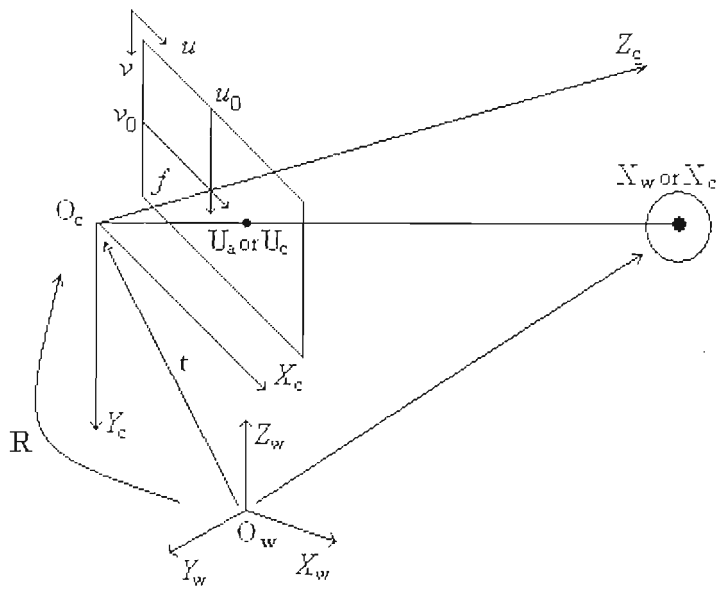


Figure 2.1: Perspective projection of a pinhole camera

The chapter subsequently discusses epipolar geometry and the derivation of the fundamental and essential matrices. It gives the reader some theories behind feature correspondence matching and the fundamental matrix estimation stage. This is followed by a review of the different techniques for estimating a fundamental matrix from point matches that are used in the literature.

2.2 Camera Model

Normally, perspective projection (also called central projection) is used to model cameras. Perspective projection describes image formation by a pinhole camera.

The geometry of a pinhole camera is depicted in Figure 2.1. The image plane has been reflected with respect to the X_cY_c plane in order not to have a mirrored image with negative co-ordinates. It should be noted that there are three co-ordinate systems involved:

(a) The world Euclidean co-ordinate system (X_w, Y_w, Z_w).

It has origin at point $\mathbf{O}_w$.

(b) The camera Euclidean co-ordinate system (X_c, Y_c, Z_c).

It has origin at point $\mathbf{O}_c$. We can make world co-ordinates align with camera co-ordinates by performing a Euclidean transformation, which consists of a translation **t** and rotation **R**, which is shown in Figure 2.1.

(c) The image affine co-ordinate system (u, v).

The plane uv is parallel with the plane $X_c Y_c$. Axis v is parallel with axis Y_c but the axis u may have a different orientation to the axis X_c.

A point on an object with co-ordinates $\mathbf{X}_c = (x_c, y_c, z_c)^T$ ($3D$ camera Euclidean co-ordinate system) will be imaged at some point $\mathbf{U}_c = (u_c, v_c)^T$ ($2D$ camera Euclidean co-ordinate system) in the image plane. The relationship between the two co-ordinate systems is given by:

$$u_c = f \frac{x_c}{z_c} \tag{2.1}$$

and

$$v_c = f \frac{y_c}{z_c} \tag{2.2}$$

where f (focal length) is the distance between the origin $\mathbf{O}_c$ and the centre of the image plane.

In homogeneous co-ordinates, this can be written as:

$$\begin{bmatrix} su_c \\ sv_c \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.3}$$

where here $s \neq 0$ is a scale factor and in this case $s = z_c$.

Now, we need to transform the $\mathbf{U}_c = (u_c, v_c)^T$ point (expressed in 2D camera Euclidean co-ordinate system) into the $\mathbf{U}_a = (u_a, v_a)^T$ point (expressed in 2D image affine co-ordinate system), with respect to the origin in the top left-hand corner of the image plane (refer Figure 2.1). Equation (2.4) describes the transformation:

$$u_a = u_0 + \frac{u_c}{w} \tag{2.4}$$

and

$$v_a = v_0 + \frac{v_c}{h} \tag{2.5}$$

where $\mathbf{U}_{a0} = (u_0, v_0)$ is the centre co-ordinate of the image plane in pixels, w and h is the distance between adjacent pixels in the horizontal and vertical directions of the image plane respectively.

By substituting Equation (2.1) into Equation (2.4) and Equation (2.2) into Equation (2.5) and subsequently multiplying by z_c, we obtain:

$$z_c u_a = z_c u_0 + \frac{x_c f}{w} \tag{2.6}$$

and

$$z_c v_a = z_c v_0 + \frac{y_c f}{h} \tag{2.7}$$

Therefore, the transformation from a point in 3D camera Euclidean co-ordinate system to a point in 2D image affine co-ordinate system can be expressed using a 3×4 matrix as in Equation (2.8):

$$\begin{bmatrix} su_a \\ sv_a \\ s \end{bmatrix} = \begin{bmatrix} \frac{f}{w} & b & u_0 & 0 \\ 0 & \frac{f}{h} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.8}$$

where b represents shear, which is the degree of slant of the co-ordinate axis u in the camera image plane. The parameter b is normally introduced for the case when a non-pinhole camera such as a digitizing camera is used, where the image plane may not be perpendicular to the principal axis of the digitizing camera. This is measured in pixels, and usually the effect is very minimal $(b \approx 0)$ and can be neglected for simplicity. In this equation, scale factor s has the value of z_c. If s is assumed to be equal to one, Equation (2.8) can be expressed as:

$$\tilde{\mathbf{U}}_a = \left[\mathbf{K}|0\right]\tilde{\mathbf{X}}_c \tag{2.9}$$

where $\tilde{\mathbf{U}}_a$ represents a point in the 2D camera affine co-ordinate system expressed in homogeneous co-ordinates, $\mathbf{K}$ is the camera calibration matrix and $\tilde{\mathbf{X}}_c$ represents a point in the 3D camera Euclidean co-ordinate system expressed in homogeneous co-ordinates.

Let $\alpha_u = \frac{f}{w}$ and $\alpha_v = \frac{f}{h}$, then u_0, v_0, α_u and α_v, the coefficients of matrix $\mathbf{K}$ are called the intrinsic parameters of the camera (assuming $b = 0$). These four coefficients describe the camera, independent of its position and orientation in space.

If the position and orientation of the camera in space is considered, then Equation (2.9) will become:

$$\tilde{\mathbf{U}}_a = \left[\mathbf{K}\middle|0\right]\mathbf{M}\tilde{\mathbf{X}}_w \tag{2.10}$$

where $\tilde{\mathbf{X}}_w$ represents a point in 3D world Euclidean co-ordinate system expressed in homogeneous co-ordinates and $\mathbf{M}$ is a 4×4 transformation matrix:

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{R} & -\mathbf{Rt} \\ 0_3^t & 1 \end{array}\right] \tag{2.11}$$

In Equation (2.11), $\mathbf{R}$ represents a 3×3 rotation matrix that encodes camera orientation with respect to a given world frame and 3×1 vector $\mathbf{t}$ captures the camera's displacement from the world frame origin. The matrix $\mathbf{R}$ can be expressed as a function of ϕ, θ, and φ as follows:

$$\mathbf{R} = \begin{bmatrix} \cos\varphi\cos\theta & \sin\varphi\cos\theta & -\sin\theta \\ -\sin\varphi\cos\phi + \cos\varphi\sin\theta\sin\phi & \cos\varphi\cos\phi + \sin\varphi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\varphi\sin\phi + \cos\varphi\sin\theta\cos\phi & -\cos\varphi\sin\phi + \sin\varphi\sin\theta\cos\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.12}$$

where ϕ, θ, and φ are the angle of rotation about the X_w, Y_w and Z_w axis respectively. The translation vector $\mathbf{t}$ is:

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.13}$$

which represents the displacement between point $\mathbf{O}_w$ and $\mathbf{O}_c$ in 3D space.

As we can see, matrix $\mathbf{M}$ has six degrees of freedom, three for the orientation, and three for the translation of the camera. These parameters are known as the extrinsic camera parameters.

After combining 3×4 camera calibration matrix $\left[\mathbf{K}|0\right]$ and the 4×4 transformation matrix $\mathbf{M}$ in Equation (2.10), we obtain a single 3×4 matrix $\mathbf{P}$, which is called the projective matrix, as shown in Equation (2.14):

$$\tilde{\mathbf{U}}_a = \mathbf{P}\tilde{\mathbf{X}}_w \tag{2.14}$$

where the general form of $\mathbf{P}$ can be expressed as a function of $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$, as shown in Equation (2.15):

$$\mathbf{P} = \left[\mathbf{KR}|-\mathbf{KRt}\right] \tag{2.15}$$

Generally, Equation (2.14) and Equation (2.15) show the resulting matrix $\mathbf{P}$ derived from an ideal pinhole camera model that relates the 3D co-ordinates of the real world with the 2D co-ordinates of the image plane. Once this relation is found, the next step is to find a means to determine the intrinsic and extrinsic parameters.

2.2.1 Algorithm for Calibrating a Camera from a Known Scene

In order to calibrate a camera from a known scene, a calibration grid is often used. The calibration process will normally require two stages. First, the estimation of matrix $\mathbf{P}$ and second, the estimation of intrinsic and extrinsic parameters from the matrix $\mathbf{P}$. The first stage can be achieved by finding 3D points from the calibration grid and their corresponding 2D points in the image plane. The process can be described as follows:

Let p_{ij} represents the entry of matrix $\mathbf{P}$ at i^{th} row and j^{th} column and where $1 \le i \le 3$ and $1 \le j \le 4$, then Equation (2.14) can be rewritten as follows:

$$\begin{bmatrix} su_a \\ sv_a \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} su_a \\ sv_a \\ s \end{bmatrix} = \begin{bmatrix} p_{11}x_c + p_{12}y_c + p_{13}z_c + p_{14} \\ p_{21}x_c + p_{22}y_c + p_{23}z_c + p_{24} \\ p_{31}x_c + p_{32}y_c + p_{33}z_c + p_{34} \end{bmatrix} \tag{2.17}$$

$$\begin{aligned} u_a(p_{31}x_c + p_{32}y_c + p_{33}z_c + p_{34}) &= p_{11}x_c + p_{12}y_c + p_{13}z_c + p_{14} \\ v_a(p_{31}x_c + p_{32}y_c + p_{33}z_c + p_{34}) &= p_{21}x_c + p_{22}y_c + p_{23}z_c + p_{24} \end{aligned} \tag{2.18}$$

Equations (2.18) shows that for each known corresponding scene and image point, we get two linear equations with 12 unknowns $p_{11},\ldots,p_{34}$. If there are n correspondences, Equations (2.18) can be written as a $2n \times 12$ matrix,

$$\begin{bmatrix} x_{c1} & y_{c1} & z_{c1} & 1 & 0 & 0 & 0 & 0 & -u_{a1}x_{c1} & -u_{a1}y_{c1} & -u_{a1}z_{c1} & -u_{a1} \\ 0 & 0 & 0 & 0 & x_{c1} & y_{c1} & z_{c1} & 1 & -v_{a1}x_{c1} & -v_{a1}y_{c1} & -v_{a1}z_{c1} & -v_{a1} \\ & & & & & & \vdots & & & & & \\ x_{cn} & y_{cn} & z_{cn} & 1 & 0 & 0 & 0 & 0 & -u_{an}x_{cn} & -u_{an}y_{cn} & -u_{an}z_{cn} & -u_{an} \\ 0 & 0 & 0 & 0 & x_{cn} & y_{cn} & z_{cn} & 1 & -v_{an}x_{cn} & -v_{an}y_{cn} & -v_{an}z_{cn} & -v_{an} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ \vdots \\ p_{33} \\ p_{34} \end{bmatrix} = 0 \tag{2.19}$$

Based on Equation (2.19), at least six known 3D to 2D correspondence points are needed to obtain a solution. In the event of more than six correspondence points, which is always the case, Equation (2.19) is solved using the least-square method.

After matrix $\mathbf{P}$ is solved, the extrinsic parameters (the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$) can be obtained by rewriting projective matrix $\mathbf{P}$ as Equation (2.20).

$$\mathbf{P} = \left[\mathbf{KR} \middle| -\mathbf{KRt}\right] = \left[\mathbf{A} \middle| \mathbf{b}\right] \tag{2.20}$$

If $\mathbf{A} = \mathbf{KR}$ and $\mathbf{b} = -\mathbf{KRt}$ then translation vector $\mathbf{t}$ can be obtained by $\mathbf{t} = (-\mathbf{KR})^{-1}\mathbf{b} = -\mathbf{A}^{-1}\mathbf{b}$. Rotation matrices $\mathbf{R}$ and $\mathbf{K}$ can be found by using matrix factorization, namely QR decomposition [Press et al. 1992; Golub and Loan 1989] which will decompose matrix $\mathbf{A}$ into a product of upper triangular matrix and orthogonal matrix. Note that the rotation matrix is an orthogonal matrix and calibration matrix $\mathbf{K}$ is an upper triangular matrix.

2.3 Lens Distortion

This subsection will give reader some theoretical background on lens distortion in order to further understand Chapter 5, which will discuss the

integration of distortion correction in camera self-calibration for AR. This subsection describes different types of lens distortion and how to model it mathematically.

In theory, the lens performs an ideal central projection, but with real lenses, this is not the case. A typical lens performs with distortion of several pixels. Tordoff and Murray [2000] describe distortion as the displacement of an image point from the position predicted by the ideal pinhole camera model. There are two components of lens distortion [Brown 1971; Slama 1980]:

1) Radial distortion, which bends the ray more or less than in the ideal case. This is usually caused by an imperfect lens shape [Zhang 1996]. The ideal image points are distorted along radial directions from the distortion centre (normally the principal point).
2) De-centring, which displaces the principal point from the optical axis. This is caused by improper lens assembly [Zhang 1996].

Lens distortion can be described by the following equations:

$$\hat{u}_c = u_c + \delta_u \tag{2.21}$$

and

$$\hat{v}_c = v_c + \delta_v \tag{2.22}$$

where (u_c, v_c) are the distorted (true) image co-ordinates on the image plane and (δ_u, δ_v) are the distortion corrections to $(\hat{u}_c, \hat{v}_c)$. The distortions are often modelled as even power polynomials in order to secure rotational symmetry as shown in Equation (2.23) and (2.24):

$$\delta_u = u_c(k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) + [p_1(r^2 + 2u_c^2) + 2p_2 u_c v_c](1 + p_3 r^2 + \ldots) \tag{2.23}$$

$$\delta_v = v_c(k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) + [2p_1 u_c v_c + p_2(r^2 + 2v_c^2)](1 + p_3 r^2 + \ldots) \tag{2.24}$$

where $r^2 = u_c^2 + v_c^2$, k_1, k_2 and k_3 are coefficients of radial distortion; p_1, p_2 and p_3 are coefficients of de-centring distortion.

Distortions are usually modelled depending on the type of lenses used. According to Tsai [1987] and Brown [1971], unless one is specifically concerned with the reduction of distortion to very low levels, it is likely that the distortion function is totally dominated by the k_1 term. The first two radial terms are enough to model distortion since any more elaborate modelling not only would not help (negligible when compared with sensor quantisation), but also would cause numerical instability. This is confirmed by Wei and Ma [1994]. To avoid any cause of instability to our self-calibration algorithm, we will only use the k_1 term to model the distortion.

Figure 2.2 shows examples of radial distortion. On the left is pincushion-like distortion (a minus sign in Equation (2.25) and (2.26)), and on the right a depiction of barrel-like distortion corresponding to a plus sign.

Figure 2.2: Example of radial distortion.

$$\hat{u}_c = u_c[1 \pm k_1(u_c^2 + v_c^2)] \tag{2.25}$$

$$\hat{v}_c = v_c[1 \pm k_1(u_c^2 + v_c^2)] \tag{2.26}$$

2.4 Epipolar Geometry

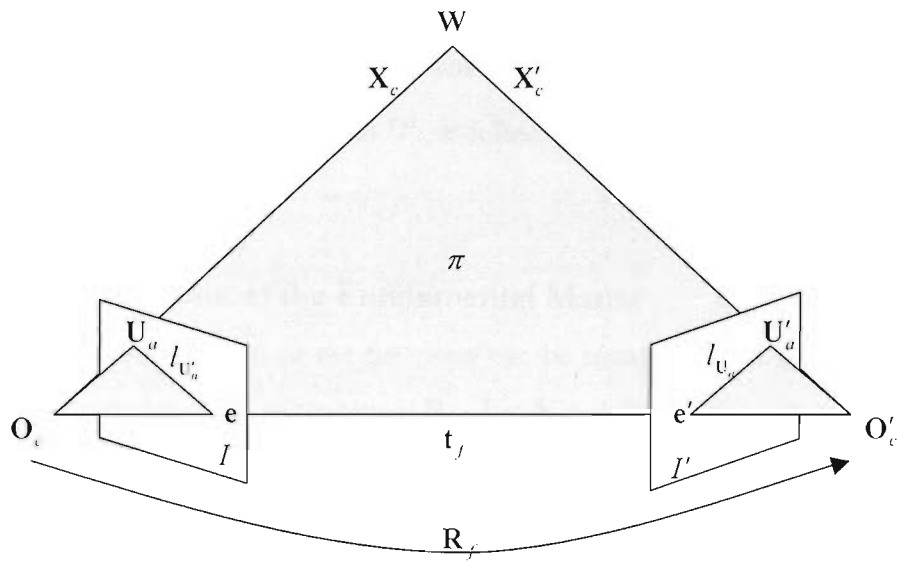


Figure 2.3: Epipolar Geometry

This subsection describes background theories in epipolar geometry and how the essential matrix and fundamental matrix are derived from the vision perspective. This will help the reader to understand how the constraints of the algebraic approach for self-calibration based on three views are derived, as discussed in Chapter 5.

Epipolar geometry exists between any two camera systems or a single camera with two views separated by general motion. Consider the case of two cameras as shown in Figure 2.3.

Let $\mathbf{O}_c$ and $\mathbf{O}'_c$ be the optical centres of the first and second cameras, respectively, and let I and I' be the image planes of the first and second cameras respectively. Given a point $\mathbf{U}_a$ in the first image, its corresponding point in the second image is constrained to lie on a line called the epipolar

line of $\mathbf{U}_a$, denoted by $l_{\mathbf{U}_a}$. Similarly, point $\mathbf{U}'_a$ in the second image has its corresponding point constrained to lie on epipolar line $l_{\mathbf{U}'_a}$. The line $l_{\mathbf{U}_a}$ is the intersection of the epipolar plane π, defined by $\mathbf{W}$, $\mathbf{O}_c$ and $\mathbf{O}'_c$ with second image plane I'. Similarly, the line $l_{\mathbf{U}'_a}$ is the intersection of epipolar plane π, defined by $\mathbf{W}$, $\mathbf{O}_c$ and $\mathbf{O}'_c$ with first image plane I.

2.4.1 Derivation of the Fundamental Matrix

The co-ordinate system of the first view can be transformed to the right view by translation $\mathbf{t}_f$ and rotation $\mathbf{R}_f$. Let $\mathbf{K}$ and $\mathbf{K}'$ be the calibration matrices of the left and right cameras respectively. Let the centre of the world Euclidean co-ordinate system be aligned with the centre of the first camera. Based on Equation (2.9) and Equation (2.10), the left projection $\mathbf{U}_a$ and the right projection $\mathbf{U}'_a$ of the scene point $\mathbf{W}$ are shown below:

$$\mathbf{U}_a \cong \begin{bmatrix} \mathbf{K} & | & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X}_c \\ 1 \end{bmatrix} = \mathbf{K}\,\mathbf{X}_c \tag{2.27}$$

$$\mathbf{U}'_a \cong \begin{bmatrix} \mathbf{K}'\mathbf{R}_f & | & -\mathbf{K}'\mathbf{R}_f\mathbf{t}_f \end{bmatrix} \begin{bmatrix} \mathbf{X}_c \\ 1 \end{bmatrix} = \mathbf{K}'\left(\mathbf{R}_f\mathbf{X}_c - \mathbf{R}_f\mathbf{t}_f\right) = \mathbf{K}'\mathbf{X}'_c \tag{2.28}$$

Note that vectors $\mathbf{X}_c$, $\mathbf{X}'_c$ and $\mathbf{t}_f$ are co-planar and the symbol $\cong$ is used to denote projection up to unknown scale. Let subscripts $_L$ and $_R$ represent the left and right camera co-ordinate systems respectively so that $\mathbf{X}_{cL}$ and $\mathbf{X}'_{cL}$ are the co-ordinate vectors expressed with respect to the left camera co-ordinate system, while $\mathbf{X}_{cR}$ and $\mathbf{X}'_{cR}$ are the co-ordinate vectors expressed with respect to the right camera co-ordinate system. The

relationship between $\mathbf{X}'_{cR}$ and $\mathbf{X}'_{cL}$ in terms of rotation can be expressed as $\mathbf{X}'_{cR} = \mathbf{R}_f \mathbf{X}'_{cL}$, and hence $\mathbf{X}'_{cL} = \mathbf{R}_f^{-1} \mathbf{X}'_{cR}$.

If points $\mathbf{U}_a$ and $\mathbf{U}'_a$ correspond to a single physical point $\mathbf{W}$ in space, then $\mathbf{U}_a$, $\mathbf{U}'_a$, $\mathbf{O}_c$ and $\mathbf{O}'_c$ must lie in a single plane. This is called the co-planarity constraint and can be written as:

$$\mathbf{X}_{cL}^T (\mathbf{t}_f \times \mathbf{X}'_{cL}) = 0 \tag{2.29}$$

From Equation (2.27) we have $\mathbf{X}_{cL} = \mathbf{K}^{-1}\mathbf{U}_a$, $\mathbf{X}'_{cR} = (\mathbf{K}')^{-1}\mathbf{U}'_a$, and hence $\mathbf{X}'_{cL} = \mathbf{R}_f^{-1}(\mathbf{K}')^{-1}\mathbf{U}'_a$. Substituting these equations into (2.29) we have:

$$(\mathbf{K}^{-1}\mathbf{U}_a)^T (\mathbf{t}_f \times \mathbf{R}_f^{-1}(\mathbf{K}')^{-1}\mathbf{U}'_a) = 0 \tag{2.30}$$

Note that Equation (2.30) is homogeneous with respect to $\mathbf{t}_f$, meaning that scale is undetermined and the absolute scale of the scene can only be recovered if we have some extra information such as knowing the distance between two points in space.

Let $\mathbf{A}$ be a regular matrix and $\mathbf{t}_f = \left[t_x, t_y, t_z\right]^T$, then vector product $\mathbf{t} \times \mathbf{A}$ can be replaced with $\mathbf{S}(\mathbf{t}_f)\mathbf{A}$ where

$$\mathbf{S}(\mathbf{t}_f) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{2.31}$$

is a skew symmetric matrix and $\mathbf{t}_f \neq 0$.

Therefore, Equation (2.30) can be rewritten as:

$$(\mathbf{K}^{-1}\mathbf{U}_a)^T(\mathbf{S}(\mathbf{t}_f)\mathbf{R}_f^{-1}(\mathbf{K}')^{-1}\mathbf{U}'_a) = 0 \tag{2.32}$$

The fundamental matrix $\mathbf{F}$ between the two views can be obtained by rearranging Equation (2.32) to the following form:

$$\mathbf{U}_a^T\mathbf{F}\mathbf{U}'_a = 0 \tag{2.33}$$

where

$$\mathbf{F} = (\mathbf{K}^{-1})^T\mathbf{S}(\mathbf{t}_f)\mathbf{R}_f^{-1}(\mathbf{K}')^{-1} \tag{2.34}$$

It is shown in Equation (2.34) that all the information that can be recovered from a pair of images is contained in a single 3×3 matrix $\mathbf{F}$ if the correspondence problem is solved. Note that in order for point $\mathbf{U}_a$ (a point in I) to be the corresponding point for $\mathbf{U}'_a$ (a point in I'), Equation (2.33) must be satisfied. From epipolar constraints (Equation (2.33)), the search for correspondence can be reduced from searching through the whole image to a searching a single epipolar line.

2.4.2 Derivation of the Essential Matrix

The term essential matrix describes the relative motion of a single camera moving in space with known calibration, meaning that $\mathbf{K}$ has been determined. For a system with two cameras, both $\mathbf{K}$ and $\mathbf{K}'$ are known. Knowing these values allows us to normalize measurement for the left and right images. Let $\breve{\mathbf{U}}_a$ and $\breve{\mathbf{U}}'_a$ be the normalized measurements for the left and right images respectively, thus we can have the following relations:

$$\breve{\mathbf{U}}_a = \mathbf{K}^{-1}\mathbf{U}_a \tag{2.35}$$

and

$$\breve{\mathbf{U}}'_a = (\mathbf{K}')^{-1}\mathbf{U}_a \tag{2.36}$$

If we substitute Equation (2.35) and (2.36) in (2.32), we get

$$\breve{\mathbf{U}}_a^T \mathbf{S}(\mathbf{t}_f)\mathbf{R}_f^{-1}\breve{\mathbf{U}}'_a = 0 \tag{2.37}$$

Essential matrix $\mathbf{E}$ can be obtained by substituting $\mathbf{E} = \mathbf{S}(\mathbf{t}_f)\mathbf{R}_f^{-1}$ and thus we get the following:

$$\breve{\mathbf{U}}_a^T \mathbf{E}\,\breve{\mathbf{U}}'_a = 0 \tag{2.38}$$

It is shown in Equation (2.37) that essential matrix $\mathbf{E}$ contains all the information about the relative motion from the first to the second position of a calibrated camera.

2.4.3 Essential Matrix and Fundamental Matrix Properties

The properties of the essential matrix can be listed as follows [Sonka et al. 1999]:

- The essential matrix $\mathbf{E}$ has rank 2.
- Let $\mathbf{t}_f$ be the translational vector, and $\mathbf{c} = \mathbf{R}_f\mathbf{t}_f$. Then $\mathbf{Ec} = 0$ and $\mathbf{c}^T\mathbf{E} = 0$.
- SVD (singular value decomposition) decomposes $\mathbf{E}$ as $\mathbf{E} = \mathbf{UDV}^T$ for a diagonal $\mathbf{D}$ (note that $\mathbf{U}$ is not the same as $\mathbf{U}_a$); then

$$\mathbf{D} = \begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.39}$$

- Equation (2.38) can be rearranged from

$$\breve{\mathbf{U}}_a^T \mathbf{S}(\mathbf{t}_f)\mathbf{R}_f^{-1}\breve{\mathbf{U}}_a' = 0$$

to

$$\breve{\mathbf{U}}_a'^T \mathbf{R}_f \mathbf{S}(\mathbf{t}_f)\breve{\mathbf{U}}_a = 0 \tag{2.40}$$

where $\mathbf{E}$ can now be represented as $\mathbf{E} = \mathbf{R}_f \mathbf{S}(\mathbf{t}_f)$. Using SVD, we can calculate the rotation matrix $\mathbf{R}_f$ and translation $\mathbf{t}_f$ between two views when essential matrix $\mathbf{E}$ has already been estimated. The procedure is as follows [Hartley 1992]:

SVD decomposes essential matrix $\mathbf{E}$ as the product of $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$. Let $\mathbf{G}$ and $\mathbf{Z}$ be defined as:

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.41}$$

$$\mathbf{Z} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.42}$$

The rotation matrix $\mathbf{R}_f$ can be calculated as

$$\mathbf{R}_f = \mathbf{U}\mathbf{G}\mathbf{V}^T \text{ or } \mathbf{R}_f = \mathbf{U}\mathbf{G}^T\mathbf{V}^T \tag{2.43}$$

and $\mathbf{S}(\mathbf{t}_f)$ from Equation (2.40) can be obtained from

$$\mathbf{S}(\mathbf{t}_f) = \mathbf{V}\,\mathbf{Z}\,\mathbf{V}^T \tag{2.44}$$

The properties of the fundamental matrix $\mathbf{F}$ can be summarized as follows:

- The fundamental matrix $\mathbf{F}$ has rank 2.
- SVD of the fundamental matrix gives $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where

$$\mathbf{D} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad k_1 \neq k_2 \neq 0 \text{ and } k_1 > k_2 \tag{2.45}$$

- Let $\mathbf{e}$ and $\mathbf{e}'$ be the epipoles in the first and second image respectively, then

$$\mathbf{e}^T\mathbf{F} = 0 \tag{2.46}$$

and

$$\mathbf{F}\,\mathbf{e}' = 0 \tag{2.47}$$

2.4.4 Techniques for Fundamental Matrix Estimation

Let a homogeneous point $\tilde{\mathbf{U}}_{ai} = [u_{ai}, v_{ai}, 1]^T$ in the first image be matched to a homogeneous point $\tilde{\mathbf{U}}'_{ai} = [u'_{ai}, v'_{ai}, 1]^T$ in the second image where i represent i^{th} point. They must satisfy the epipolar Equation (2.33):

$$[u_{ai}, v_{ai}, 1]\mathbf{F}\begin{bmatrix} u'_{ai} \\ v'_{ai} \\ 1 \end{bmatrix} = 0 \tag{2.48}$$

Rewriting the 3 x 3 fundamental matrix $\mathbf{F}$ as a column vector of 9 unknown coefficients **f**, Equation (2.48) can be written as a system of linear equations:

$$\mathbf{a}_i^T \mathbf{f} = 0 \tag{2.49}$$

where

$$\mathbf{a}_i = \left[u_{ai}u'_{ai},\ v_{ai}u'_{ai},\ u'_{ai},\ u_{ai}v'_{ai},\ v_{ai}v'_{ai},\ v'_{ai},\ u_{ai},\ v_{ai},\ 1\right]^T,$$

$$\mathbf{f} = \left[F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}\right]^T$$

and F_{ij} is the element of $\mathbf{F}$ at row i and column j.

For n point matches, we have the following linear system:

$$\mathbf{A}_n \mathbf{f} = 0 \tag{2.50}$$

where

$$\mathbf{A}_n = \left[\mathbf{a}_1, \ldots, \mathbf{a}_n\right]^T \tag{2.51}$$

These equations allow us to estimate the epipolar geometry between two views.

There are many techniques for estimating the fundamental matrix reported in the literature; some of these are listed below [Zhang 1996a]:

1. Exact solution with 7 point matches [Huang and Netravali 1994].
2. Analytic method with 8 or more point matches:
    a. Linear least-squares technique.
    b. Eigen analysis.
    c. Imposing the rank-2 constraint [Hartley 1995].
    d. Geometric interpretation of the linear criterion.
    e. Normalizing input data [Hartley 1995].

3. Analytic method with rank-2 constraint [Faugeras 1995].
4. Non-linear method minimising distances of points to epipolar lines:
    a. Iterative linear method
    b. Non-linear minimization in parameter space
5. Gradient-based technique
6. Non-linear method minimizing distances between observation and reprojection [Faugeras 1993]

2.5 Summary and Conclusions

This chapter has reviewed some of the theories necessary for a better understanding of this dissertation. The main topics addressed were:

Camera model – the camera model is based on a pinhole model. The mathematical background on how a point in a 3D world coordinate is related to a point in a 2D image plane is presented. This includes the derivation of a camera calibration matrix consisting of α_u, α_v, u_0 and v_0.

Lens distortion – the types of distortion existing in a typical camera lens were depicted, namely barrel and pincushion. This includes the derivation of the mathematical equation used to represent both types of distortion. The brief introduction provides some background to help the reader understand Chapter 5 further when distortion correction is incorporated into the AR system.

Epipolar geometry – the constraints existing between a point in one image plane with its epipolar line in a second image plane sharing the same view were presented. The fundamental matrix and essential matrix and how they were derived were briefly discussed.

In computer vision, camera calibration techniques can be classified into two cases, as follows [Sonka et al. 1999]:

1. Known scene

In this case, a set of non-coplanar points lies in the 3D world, and the corresponding image points are known. In order to achieve this, calibration objects are often used. Each 3D scene that corresponds to a 2D-image point provides one equation. The matrix coefficients are found by solving an over-determined system of linear equations [Faugeras 1993]. Even though this calibration technique gives the most accurate approximation of the intrinsic parameters, the calibration process involves a lengthy time. This condition is even worse if the focal length of the camera needs to be changed online and the camera needs to be recalibrated.

2. Unknown scene

In this case, different camera views become important since the solution of the calibration matrix can no longer depend on the calibration object. It has to depend on camera motion. From here, there are two cases:

(a) Known camera motion

According to the known motion constraints (which are rotation and translation), this can be further divided into three cases:

i. Both rotation and translation

   The general case of arbitrary known motion from one view to another view has been solved [Horaud et al. 1995].

ii. Pure rotation

   Hartley [1994] has given the general solution for the case where the camera motion is restricted to pure rotation.

iii. Pure translation

   The linear solution of pure translation is given in [Pajdla and Hlaváč 1995].

By knowing the camera motion, the coefficients of the essential matrix can be calculated and knowledge of the matrix can then be used to find the intrinsic parameters. The problem with this kind of calibration is that the known camera motion can mostly be accomplished through a certain set-up in a particular place, such as in the lab. The future direction of AR where the head mounted camera is going to be used not only indoors but also outdoors requires a more flexible approach to camera calibration.

(b) Unknown camera motion (camera self-calibration)

According to Maybank and Faugeras [1992] and Faugeras et al. [1992], in this case, the solution is non-linear and at least three views are needed. Calibration from an unknown scene is still considered numerically hard. Consideration of the problem can be found in Butterfield [1997].

Despite being numerically hard, it is believed that self-calibration will continue to be one of the important research topics in computer vision because of its flexibility. With fast development in AR applications, it is hoped that this research will make some contributions to solving a well-known problem in AR, namely the problem of registration with added flexibility through camera self-calibration.

Chapter 3

CORNER DETECTION

3.1 Introduction and Motivation

Vision-based tracking in augmented reality normally needs pre-defined marker patterns or fiducials for tracking. They are also used for reference as to where the virtual object would be overlaid in the real world. ARToolKit detects the presence of a marker in a scene by finding objects that have a square shape (refer Figure 1.4). An AR system with fiducials, on the contrary, uses different colour coding to differentiate the fiducials from other objects and to track their position in the scene. All fiducials and marker patterns need to be kept in view so that the system knows where the virtual object needs to be overlaid. When people move into markerless AR, detection of available features in the scene becomes more important. This is because the system can no longer depend on markers or fiducials for tracking, but needs to start using other features in the scene as a means of tracking and reference to register virtual objects.

If we refer to Figure 1.4, edge and corner detection of markers are already being used in ARToolKit. One might ask why we do not simply use the feature extraction algorithm contained in ARToolKit and what is the point of developing a new method of corner detection? We need to be aware that

the edge and corner detection developed in ARToolKit is designed specifically to extract edges and corners for its markers only and is not general enough to detect and extract other corners or edges in the scene. In addition, the purpose of developing new corner detection is to cater for pre-calibration stages not only for the ARToolKit system but also for other AR systems, which might not use any kind of marker or fiducial in their system (markerless environment). Furthermore, as shown in Chapter 2, the minimum number of corners needed as input for feature correspondence matching and fundamental matrix estimation stage is seven or eight, which might not be possible to achieve in the ARToolKit corner detection stage when only one marker is involved. Corners were chosen as the preferred features because we assume a corner is the easiest feature to find in a scene.

There are many corner detectors available, and these will be reviewed in the next subsection. The decision about which corner detector is to be chosen will depend on the need to tackle specific issues in the applications. In camera self-calibration, the most important thing that needs to be taken into consideration during feature extraction is the detection accuracy of features, which in our case refers to corner accuracy. This is due to the high sensitivity of the fundamental matrix estimation stage, which will produce different results for small changes in the position of the corners. Based on experiments conducted by Sojka [2003], we found that Harris and Stephens [1988] corner detector outperformed many other corner detectors in terms of higher number of true detected corners, lower numbers of false detections and lower localisation errors which means higher accuracy. For these reasons, we chose Harris corner detector as a candidate to be used in the feature detection stage.

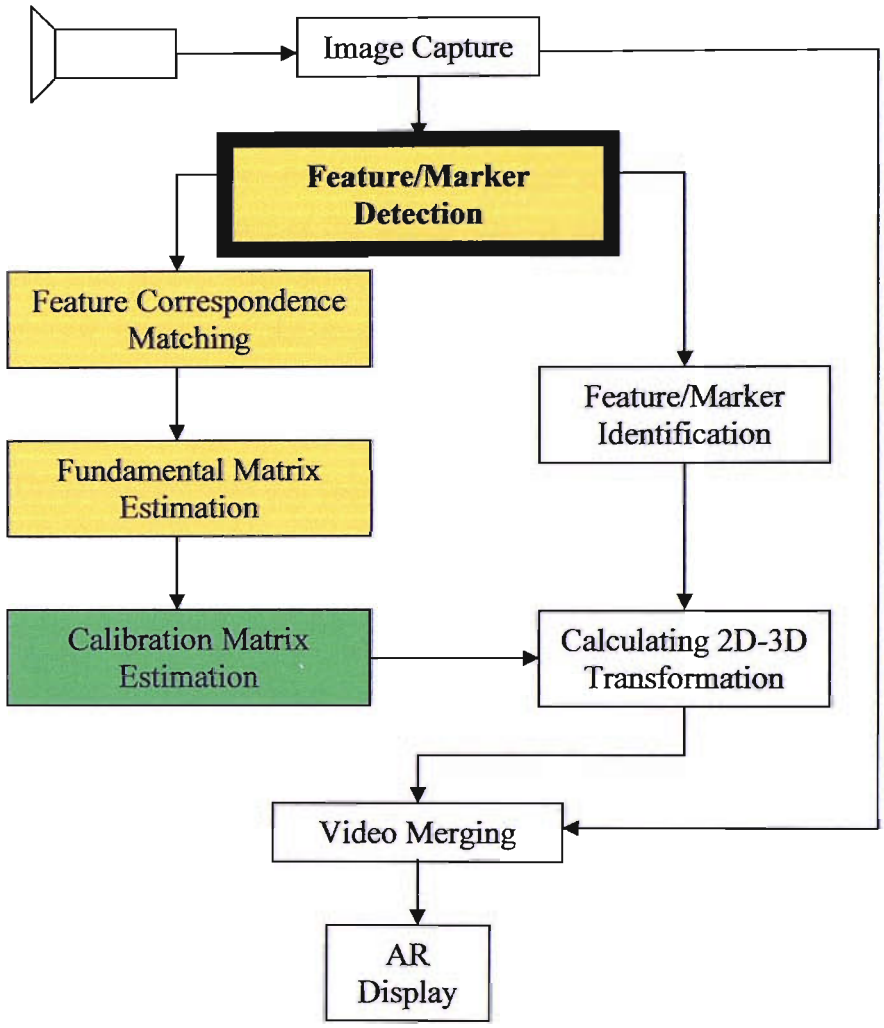


Figure 3.1: The position of the feature detection stage (corner detection) in the proposed self-calibration for an AR system.

A set of good localised corners in the AR system during the feature detection stage will help the system to perform good tracking and also help in reducing the amount of error propagation through the pre-calibration stages. This is to ensure that only the best corners are provided for the fundamental matrix estimation stage. However, the problem with good available detectors like our chosen detector Harris is that they are based on edge detectors which mean that even though corners can be successfully detected, the accuracy of the detection is not adequate. This is due to the fact that general edge detectors cannot localise edge points well around corners because of the rounding effect, leading to errors in reporting corners [Shen and Wang 2001].

The second criterion that needs to be taken into account in an AR system is that the corner detection algorithm must be fast due to its real-time application. Based on the problems of accuracy and speed, we propose a new refinement of Harris corner detector in order to improve the corner localisation. The refinement algorithm is based on the idea of calculating area in a circular mask as developed by Smith and Brady [1995] who developed the SUSAN corner detector. With this, we can benefit from the good performance of Harris corner detector and at the same time the refinement of corners will not take too much time since SUSAN algorithm was reported to be 10 times faster than the Harris corner detector [Shen and Wang 2001], apart from its robustness to noisy images [Cazorla et al. 1999]. We denote the combination of these two corner detectors algorithm as Harris-SUSAN hybrid.

In this chapter, the first pre-calibration stage, which is the corner detection stage, is presented. The stage referred to is shown in the bold box in Figure 3.1. We name it **feature/marker** to show that this proposed self-calibration can be used with or without the presence of markers or fiducials in the scene. If a marker is present, its 4 corners will be used as the input to the

next pre-calibration stages, plus other corners available in the scene. If no marker is present, then only the available corners in the scene will be the input. This is to ensure that the calibration matrix can still be updated in the presence or absence of markers.

This chapter then reviews the corner detectors currently available with brief discussions of their strengths and weaknesses. We then present more detail on the Harris and Susan corner detectors. After that, the new Harris-SUSAN hybrid corner detector is proposed. Experiments showing comparisons with original Harris and SUSAN corner detectors in terms of localisation error for different test images are presented and discussed.

3.2 Literature Review on Corner Detection

Many corner detectors have been reported in the past 20 years. Perhaps the simplest corner detector is the Moravec detector, which defines corners as points where there is a large intensity variation in every direction [Moravec 1977].

Zuniga and Haralick [1983] and Haralick and Shapiro [1992] then produced a better corner detectors than the Moravec detector, although computationally more expensive, which is based on facets, in which the neighbourhood of each image pixel is modelled as a piecewise continuous function. Once the facet model parameters have been obtained for each image pixel the response is calculated using the Zuniga-Haralick operator [Sonka et al. 1999].

Kitchen and Rosenfeld [1982] proposed a gradient-based corner detector by measuring the curvature of an edge that passes through a neighbourhood. The edge strength and the rate of change of edge direction are the measures that determine the strength of the corner response.

Freeman and Davis [1977], Asada and Brady [1986] and Medioni and Yasumoto [1987] are among those who propose techniques that use binary edge maps to find corners. The method will find edges and calculate edge curvature to locate corners. The disadvantage of this method is that it cannot accurately locate corners at junctions.

Rangarajan et al. [1989] developed a corner detector based on optimal function, which yields a maximum at the corner point when convolved with the grey level function.

Wang and Brady [1995] found that the total curvature of the grey level image is proportional to the second order directional derivative in the direction to edge normal and inversely proportional to edge strength [Shen and Wang 2001].

In Sojka [2003], comparison is made between several well-known corner detectors and in his experiments on corner detector the Harris detector outperforms the SUSAN detector in terms of more true corners, fewer false corners, less multiple detection and fewer missed corners.

3.3 SUSAN Corner Detector

SUSAN is the acronym for Smallest Univalue Segment Assimilating Nucleus. The underlying principle behind the SUSAN corner detector is basically to find the position of the pixel that gives the smallest value in terms of area of interest. Figure 3.2 shows four circular masks at different places on a simple image where the dark area represents the background and the light area is a simple object with 4 corners. Each mask has a nucleus that represents a pixel located at the centre of the mask.

As shown in Figure 3.3, when the mask is located on the edge, there will be some pixels that represent the background and some pixels that represent the front object. The area of the mask is defined as the total number of pixels inside the mask that have the same brightness value as the nucleus pixel. The area of the mask is known as 'USAN' (Univalue Segment Assimilating Nucleus). In effect, the area when the nucleus is near the edge is approximately half of the area of the mask. In addition, the area of the mask will be at its minimum when the nucleus falls near the corner and will be at its maximum when there is no edge inside the circle mask. Thus, a corner is said to be detected at the nucleus when the area is at its minimum.

The normal SUSAN circular mask encompasses 37 pixels with a radius of 4 pixels as shown in Figure 3.4. The following describes the mathematical aspect of the SUSAN corner detector.

Firstly the circular mask is convolved through all pixels in the image. For each convolution step, the brightness value for each pixel in the circular mask is compared with the one at the nucleus. This can be represented by:

$$c(\vec{r},\vec{r}_0)=\begin{cases}1 & if\left|I(\vec{r})-I(\vec{r}_0)\right|\le t\\ 0 & if\left|I(\vec{r})-I(\vec{r}_0)\right|>t\end{cases} \tag{3.1}$$

where $I(\vec{r}_0)$ is the pixel brightness value at position $\vec{r}_0$ (nucleus), $I(\vec{r})$ is pixel brightness value at position $\vec{r}$ (other than the nucleus within the mask) and t is the threshold for pixel brightness difference.

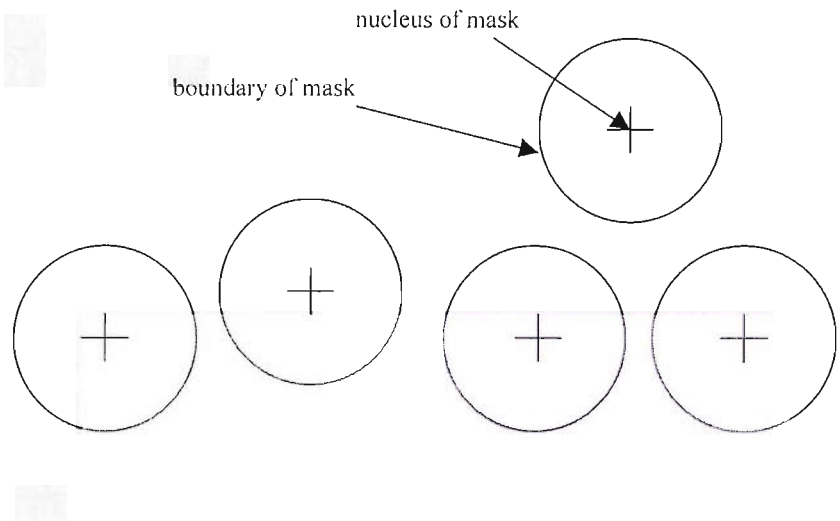


Figure 3.2: Four circular masks at different places on a simple image.

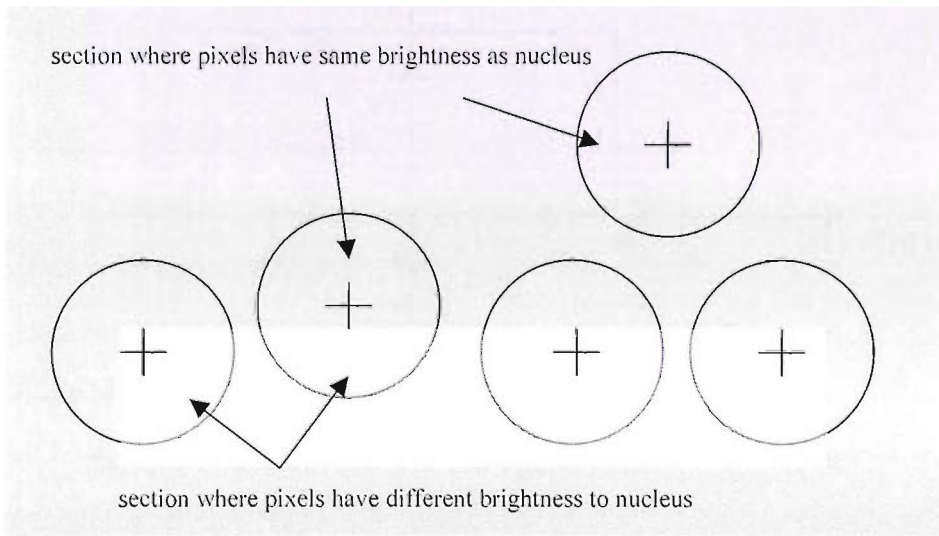


Figure 3.3: Four circular masks with similarity colouring; USANs are shown as the white parts of the masks.

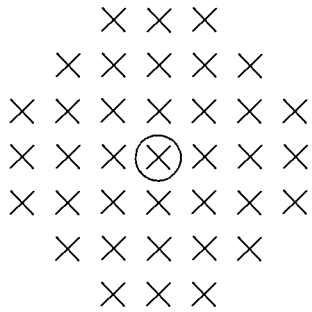


Figure 3.4: A circular mask comprised of 37 pixels with a circled cross representing the nucleus (centre).

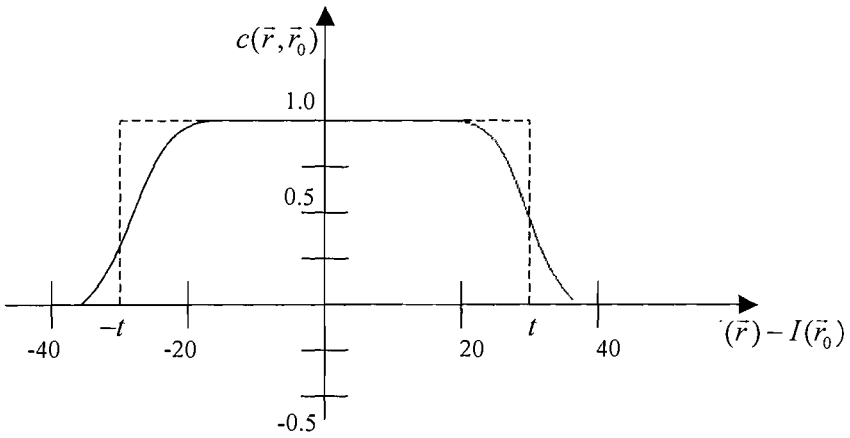


Figure 3.5: Similarity function versus pixel brightness difference.

The graph shown in Figure 3.5 comes from Equation (3.2), which is a more stable and sensible replacement for Equation (3.1).

$$c(\vec{r},\vec{r}_0) = e^{-(\frac{I(\vec{r})-I(\vec{r}_0)}{t})^6} \tag{3.2}$$

The total number of pixels in the USAN, $n(\vec{r}_0)$, is given by

$$n(\vec{r}_0) = \sum_{\vec{r}} c(\vec{r},\vec{r}_0) \tag{3.3}$$

and let geometric threshold $g = \frac{3}{4} n_{\max}$ where $n_{\max}$ is effectively the total number of pixels inside the mask, then the initial edge response can be written as:

$$R(\vec{r}_0) = \begin{cases} g - n(\vec{r}_0) & \text{if } n(\vec{r}_0) < g \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The SUSAN principle is formulated in Equation (3.4) where the edge response will be at its maximum when the USAN area is at its minimum. The value $\frac{3}{4} n_{\max}$ is created for optimal noise rejection.

3.4 Harris Corner Detector

The Harris corner detector has been used in various applications that need the reliable detection of a corner. It is also known as the Plessey corner detector and was developed by Harris and Stephen [1988]. Corners are detected by first finding the image derivatives due to the fact that the derivatives are bigger at locations where the image function undergoes rapid changes, such as around edges and corners. However, derivatives have the

effect of suppressing low frequency signals and increasing high frequency signals, which include both wanted (edges and corners) and unwanted signals (noise). To reduce the amount of noise in the image, Harris introduced a low pass Gaussian filter and to avoid the wanted signals being smoothed, the derivatives are squared. The mathematical expression of the algorithm is described in the following section.

3.4.1 Algorithm

The algorithm of the Harris corner detector can be described as follows. Let

$$\mathbf{H} = \begin{bmatrix} \overline{\left(\frac{\partial I}{\partial u}\right)^2} & \overline{\left(\frac{\partial I}{\partial u}\right)\left(\frac{\partial I}{\partial v}\right)} \\ \overline{\left(\frac{\partial I}{\partial u}\right)\left(\frac{\partial I}{\partial v}\right)} & \overline{\left(\frac{\partial I}{\partial v}\right)^2} \end{bmatrix} \tag{3.5}$$

where $I(u,v)$ is the intensity value of an image pixel. A point is detected as a corner when the two eigenvalues of the matrix are large. The sign $\bar{}$ indicates that each entry of matrix $\mathbf{H}$ is smoothed by a Gaussian filter. The corner response function Ω can be written as:

$$\Omega = \det(\mathbf{H}) - m(trace(\mathbf{H}))^2 \tag{3.6}$$

where 0.04 is the value of m as suggested by Harris, which was empirically arrived at as it gave the best result. The location of corners can be determined by extracting the local maxima of the corner response function Ω.

3.5 Harris-SUSAN Hybrid Corner Detector

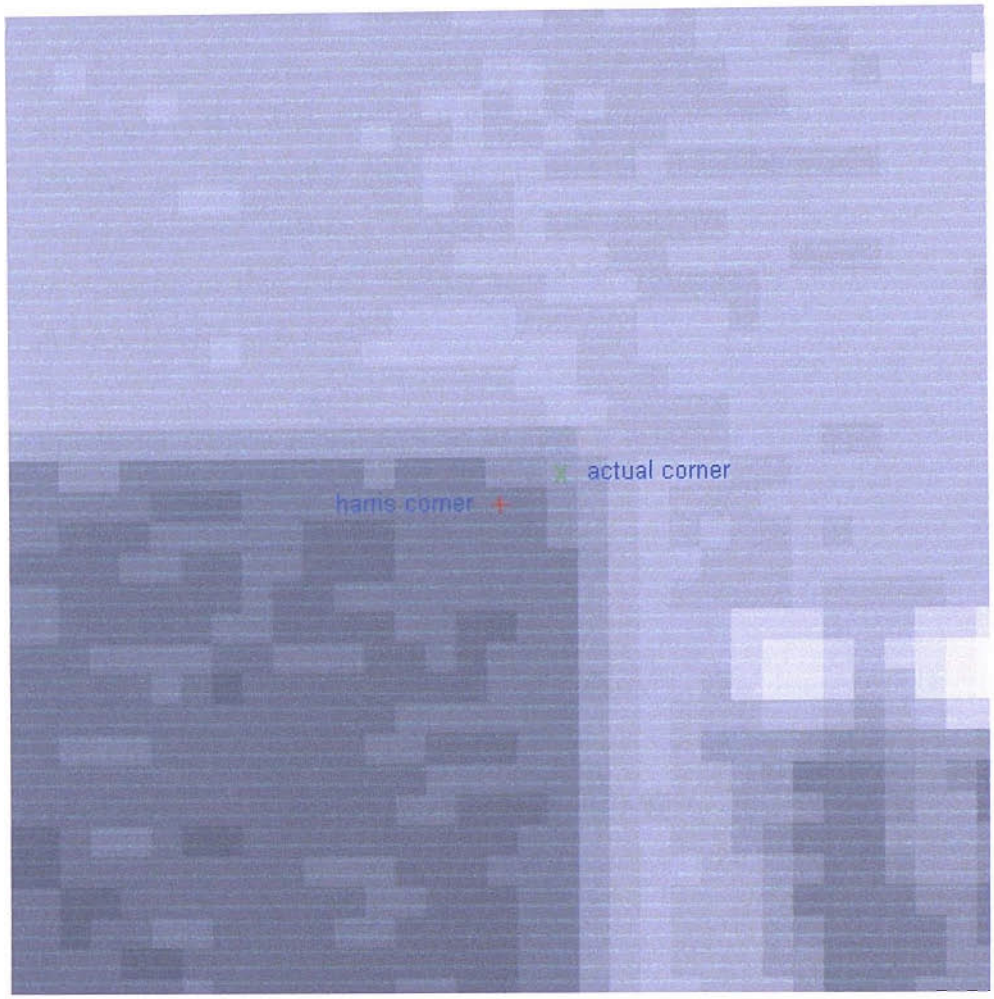


Figure 3.6: The displacement between a Harris detected corner and the actual corner.

It has been shown by Sojka [2003] that the Harris detector performs better than the SUSAN detector in terms of number of correct matches. Although the Harris detector is capable of detecting corners successfully, each individual corner detection is, however, not as accurate as it should be. The detected corners are mostly displaced between 0 to 3 pixels from the actual corner, even for a clearly strong corner. This is illustrated in Figure 3.6.

The poor localisation of corners by Harris motivates us to develop a refinement algorithm to improve the accuracy of detected corners. The algorithm uses the detected corners from Harris as a starting point and based on the calculated area in a circular mask, it will search the position of the true corners. The idea is derived from the algorithm presented by Smith and Brady [1995] who developed the SUSAN corner detector but with adaptive size of circular mask that changes based on the number of connected components in a variable window mask. We denote the new combination of Harris and SUSAN algorithm in this section as Harris-SUSAN Hybrid corner detector.

The introduction of a variable mask in SUSAN is because we believe that different sizes of mask are important to improve localisation especially in the case when two or more detected corners are very close together so that refinement to the wrong corner can be avoided.

A search window is established so that the search for true corners can be confined to a certain area. The refinement steps consist of searching for the location that has the least value of $n(\vec{r}_0)$ until the minimum value is found. For each step, the mask will be moved and compared with its neighbouring mask. These steps can be illustrated in Figure 3.7.

In Figure 3.7, the circles represent the nucleus of circular masks. At each step the values $n(\vec{r}_0)$ of masks with the nucleus positioned to the right, left, top, bottom, top-left, top-right, bottom-left and bottom-right pixel are calculated.

The mask is moved to the position where the value of $n(\vec{r}_0)$ is the lowest among the eight until it reaches the minimum.

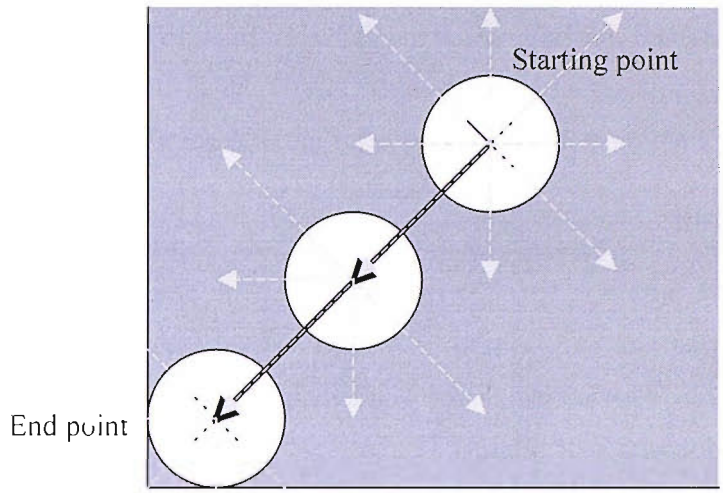


Figure 3.7: Illustration of refinement steps performed by the algorithm.

Algorithm 3.1 Refinement of Harris detected corner.

convert image into binary;

set value for threshold t;

for i = 1 to number of corners

copy selected area around corner i into a circle mask;

initialise $n(\vec{r}_0)$ by computing equation (3.3) for corner i;

while $n(\vec{r}_0)$ not minimum **do**

compute $n_t(\vec{r}_0)$, $n_b(\vec{r}_0)$, $n_l(\vec{r}_0)$, $n_r(\vec{r}_0)$, $n_{tl}(\vec{r}_0)$, $n_{tr}(\vec{r}_0)$, $n_{bl}(\vec{r}_0)$, $n_{br}(\vec{r}_0)$;

$n(\vec{r}_0) = \min(n_t(\vec{r}_0), n_b(\vec{r}_0), n_l(\vec{r}_0), n_r(\vec{r}_0), n_{tl}(\vec{r}_0), n_{tr}(\vec{r}_0), n_{bl}(\vec{r}_0), n_{br}(\vec{r}_0))$;

end while

new corner = position of $n(\vec{r}_0)$ nucleus;

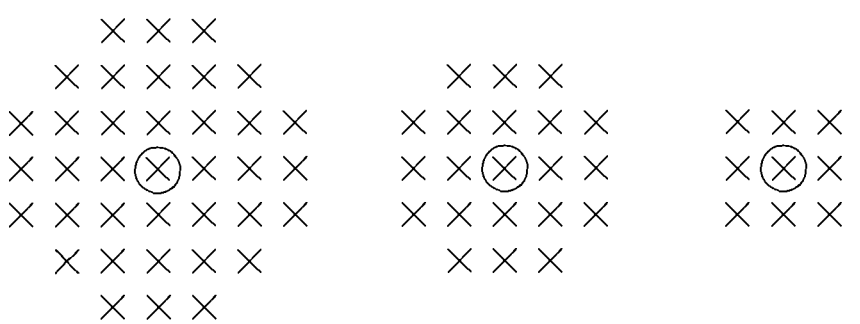
end for

Let $n_t(\vec{r}_0)$, $n_b(\vec{r}_0)$, $n_l(\vec{r}_0)$, $n_r(\vec{r}_0)$, $n_{tl}(\vec{r}_0)$, $n_{tr}(\vec{r}_0)$, $n_{bl}(\vec{r}_0)$, $n_{br}(\vec{r}_0)$ be the total number of pixels in the USAN where the nucleus position is to the top, bottom, left, right, top-left, top-right, bottom-left, bottom-right pixel of a corner. **Algorithm 3.1** explains the refinement steps.

In our algorithm we use three different sizes of mask, as shown in Figure 3.8. The size of mask changes adaptively according to the number of connected regions found in a predefined square *mask window*. The mask window size can vary between 7×7 and 5×5 pixels. The selection of the size of mask for each corner is described as follows:

1. A 7×7 mask window is positioned to a detected corner where the centre of the mask window is on the same position of the corner.
2. The number of connected region in the mask window is calculated.
3. If the number of connected region are less than or equal to 2, then the highest mask is used, otherwise the 7×7 mask window is changed into a 5×5 mask window.
4. Step 1 and 2 are repeated with a 5×5 mask window.
5. If the number of connected region in the 5×5 mask window are less than or equal to 2, then the medium mask is used, otherwise the smallest mask is selected.

The steps are illustrated in Figure 3.8 to 3.11.



Biggest mask Medium mask Smallest mask

Figure 3.8: Three different masks used in the proposed algorithm.

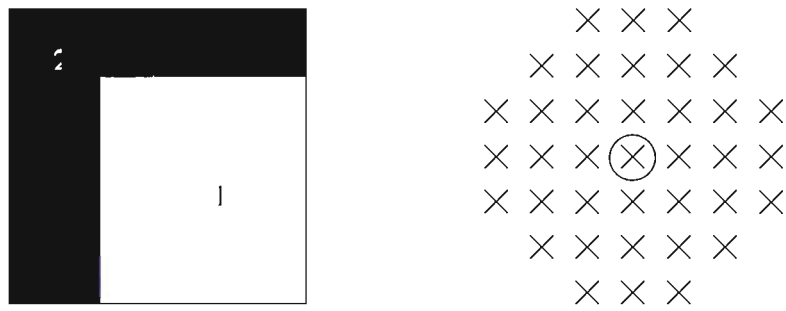


Figure 3.9: A 7×7 mask window with the number of connected regions less than or equal to two. Therefore, the biggest mask size is used.

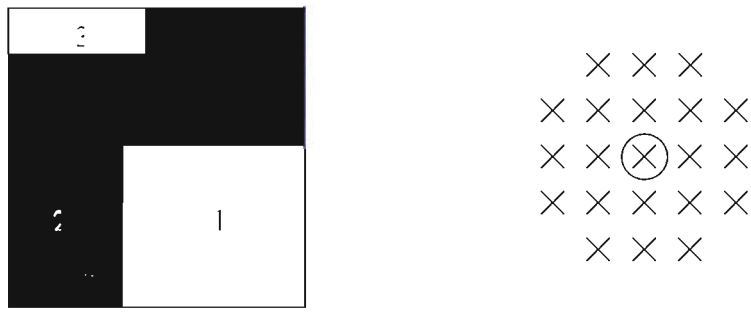


Figure 3.10: A 7×7 mask window with the number of connected regions more than two. Therefore, the medium mask size is used.

Figure 3.11: A 5×5 mask window with the number of connected regions more than two. Therefore, the smallest mask is used.

3.6 Experimental Setup

We have carried out experiments to measure the performance of our approach in terms of localisation error. In our experiments, nine test images are used. We chose different test images to measure our algorithm's performance for different AR applications, including indoor and outdoor AR. Figure 3.12(a) and Figure 3.12(b) are the test images chosen for the AR application where only the planar pattern is in the camera view. Examples of the application include positioning several marker patterns on a table, notice board or white board. Figure 3.12(c), Figure 3.12(d) and Figure 3.12(e) are chosen as representing the indoor AR cases where planar patterns *and* other objects might present at the same time. The rest of other test images (from Figure 3.12(f) until Figure 3.12(j)) represent the cases for the outdoor AR applications.

The performance in terms of localisation error of the Harris-SUSAN hybrid approach is compared with the Harris and SUSAN corner detectors.

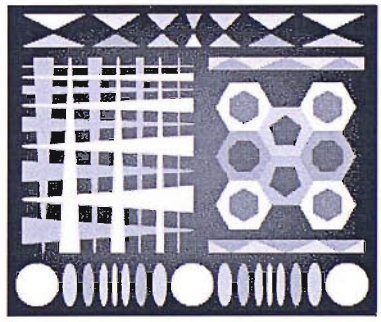


Figure 3.12(a) Pattern image 1 of size 239×200 pixels.

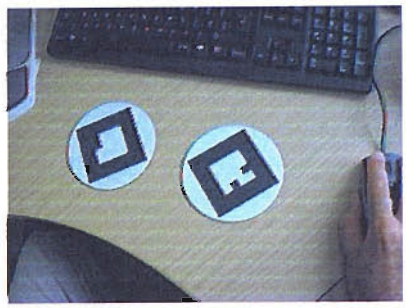


Figure 3.12(b) Pattern image 2 of size 320×240 pixels.

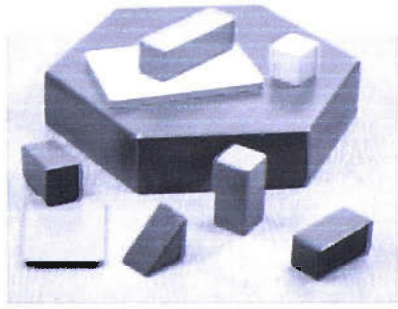


Figure 3.12(c) Box image of size 256×256 pixels.

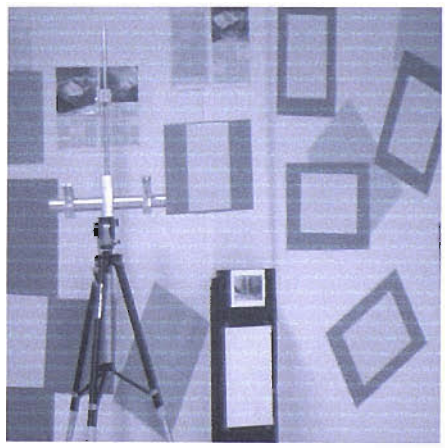


Figure 3.12(d): Lab image 1 of size 512×512 pixels.

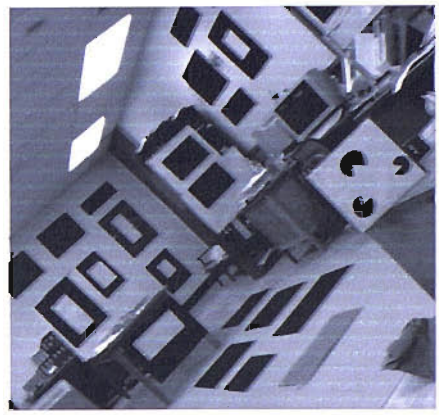


Figure 3.12(e): Lab image 2 of size 507×480 pixels.

Figure 3.12(f): Car image of size 640×440 pixels.

Figure 3.12(g): Grayscale building image of size 208×211 pixels.

Figure 3.12(h): Natural scene image of size 640×480 pixels.

Figure 3.12(i): Red building image of size 640×480 pixels.

Figure 3.12(j): Pentagon image of size 512×512 pixels.

In our implementation we set the evaluation of results as follows. Let S_{ref} and S_{est} be the set of reference solutions and set of corners found by the corner detector respectively. A corner is determined as "correct" when its distance $d(u,v)$ to the reference solution is less than Max_dist. Max_dist defines the maximum allowable distance between a corner and the reference solution for the corner to be determined as "correct". The reference solution was done manually. In our implementation we set $Max\_dist = 4$. Localisation error is defined as the average of distance $d(u,v)$ for the correct corners.

3.7 Results and Discussion

Figure 3.13 to Figure 3.18 illustrate the corners detected by Harris and our proposed Harris-SUSAN hybrid detector for different test images. We can see some improvements in corner localisation when the Harris-SUSAN hybrid detector is used in comparison with the original Harris detector.

Table 3.1: Comparison with Harris and SUSAN corner detector in terms of localisation error (in pixels) for different test images.

	Test images									
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
Harris	0.70	1.69	2.17	1.95	2.23	1.77	2.03	2.15	1.71	1.69
SUSAN	0.89	1.47	2.31	2.01	2.26	2.70	2.12	2.49	1.84	2.27
Hybrid	**0.58**	**1.16**	**2.08**	**1.63**	**2.09**	**1.64**	**1.82**	**1.87**	**1.12**	**1.63**

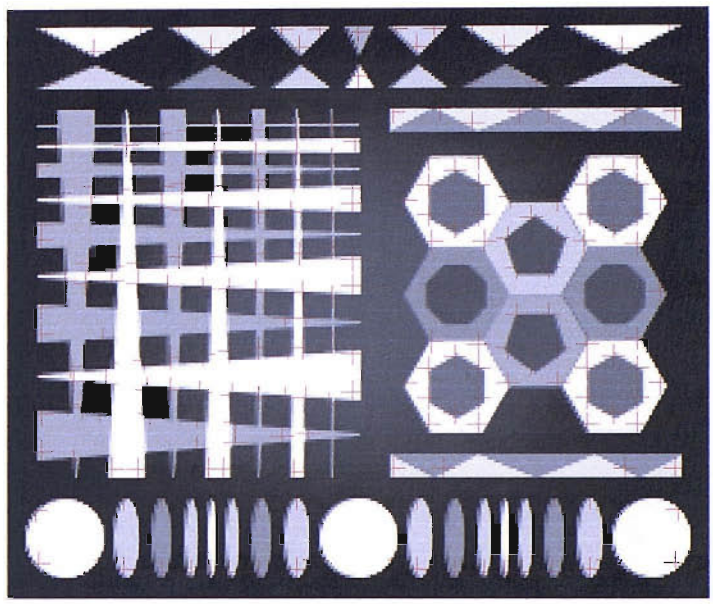


Figure 3.13: Resulting corners from Harris detector for pattern image 1.

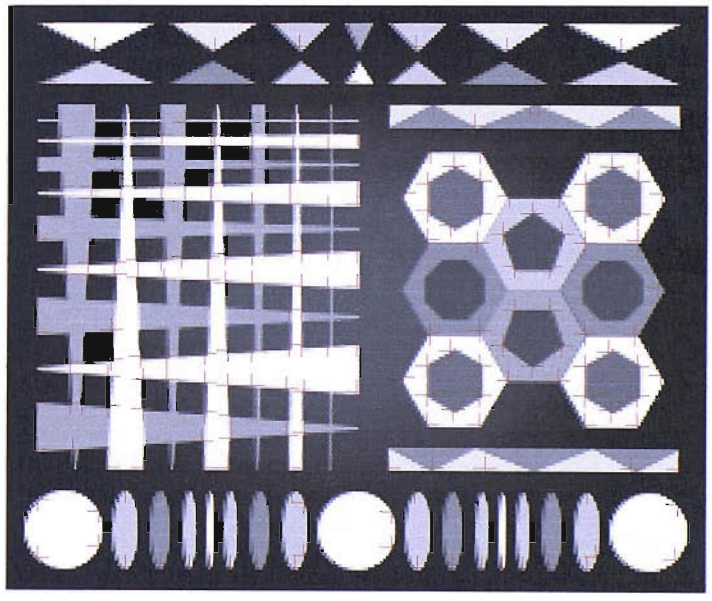


Figure 3.14: Resulting corners from Harris-SUSAN hybrid for pattern image 1

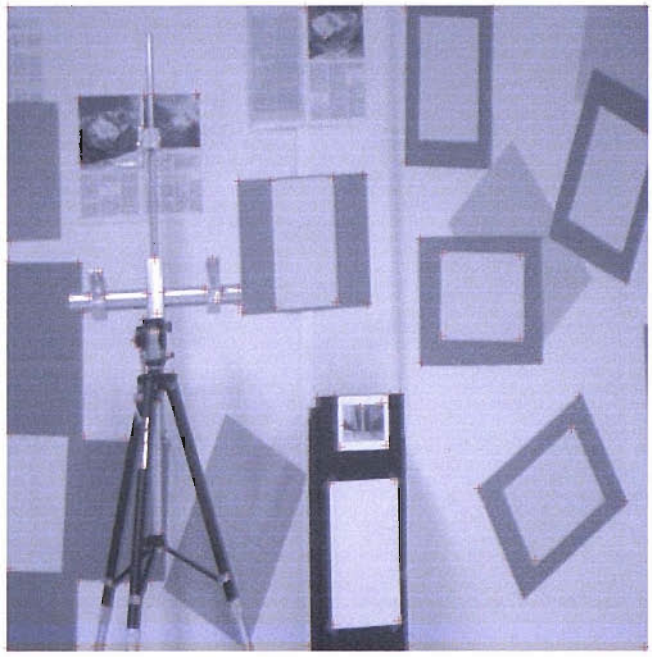


Figure 3.15: Resulting corners from Harris detector for lab image 1.

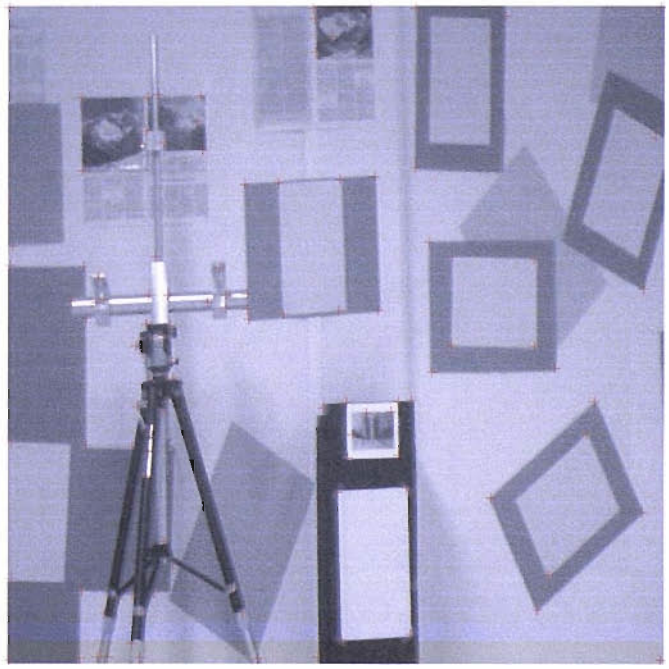


Figure 3.16: Resulting corners from Harris-SUSAN hybrid for lab image 1.

Figure 3.17: Resulting corners from Harris detector for building image.

Figure 3.18: Resulting corners from Harris-SUSAN Hybrid for building image

The comparisons between our technique and other detectors in terms of localisation errors are summarised in Table 3.1. From Table 3.1 and Figure 3.15 to Figure 3.19, the Harris-SUSAN Hybrid shows superiority over the two detectors in terms of localisation errors. However, as shown in Figure 3.20, the percentage improvement of the Harris-SUSAN Hybrid over the two detectors is varied from test image (a) to test image (j). This might be due to the different complexity of the images and because it is hard to manually determine the corner reference correctly, especially for images that contain natural objects such as trees, clouds and sky.

Another experiment carried out is to find whether the new corners resulting from the Harris-SUSAN Hybrid really improve intrinsic parameters estimation. Both sets of corners from the Harris and the Harris-SUSAN Hybrid are used to find the camera intrinsic parameters. A RANSAC algorithm [Fishler and Boles 1981] is used to find the fundamental matrix and correspondence matching technique (Chapter 3) is employed to find the correspondence for both sets of corners. As expected, Figure 3.21 clearly shows improvements in terms of the accuracy of intrinsic parameters.

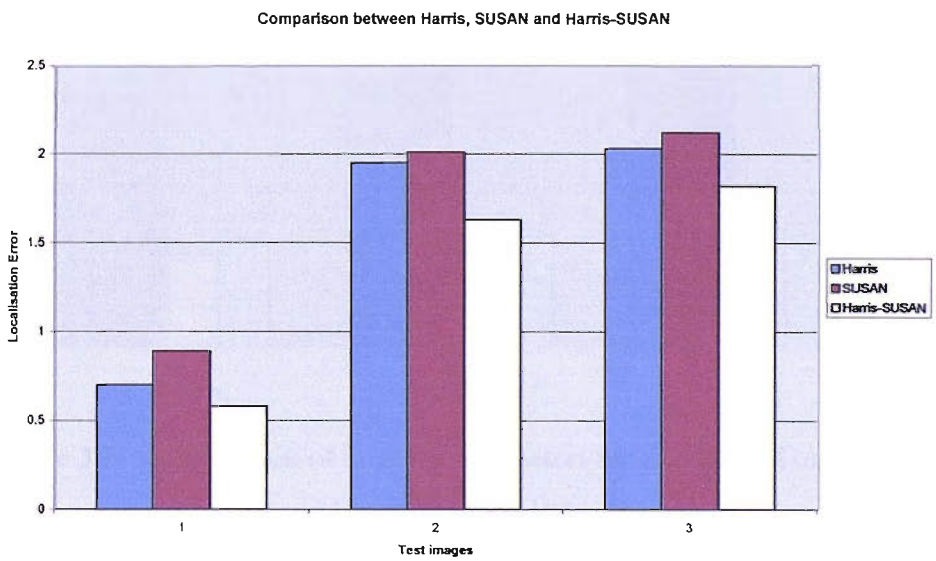


Figure 3.19: Performance in terms of localisation error for Harris, SUSAN and Harris-SUSAN hybrid detector.

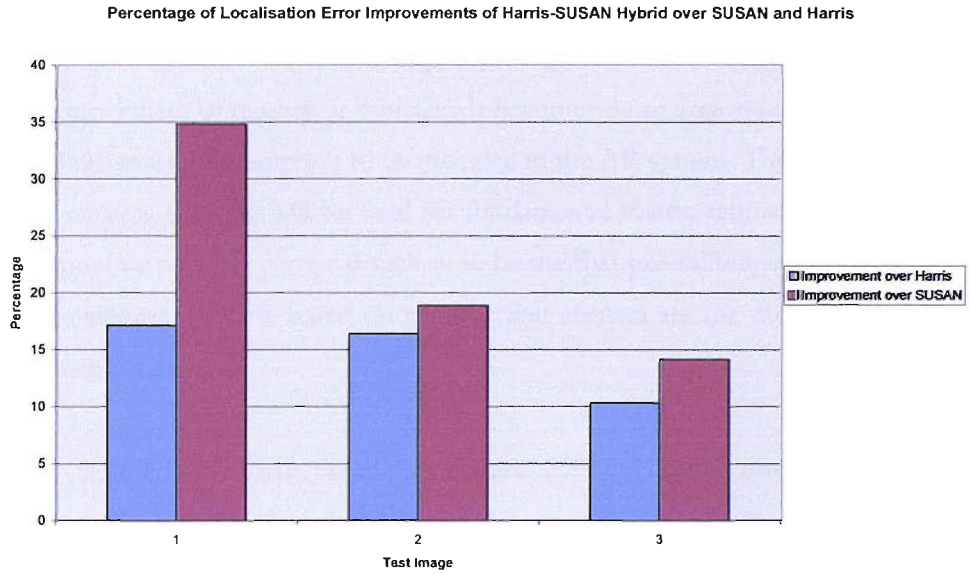


Figure 3.20: Localisation Error improvements of Harris-SUSAN Hybrid over SUSAN and Harris detector.

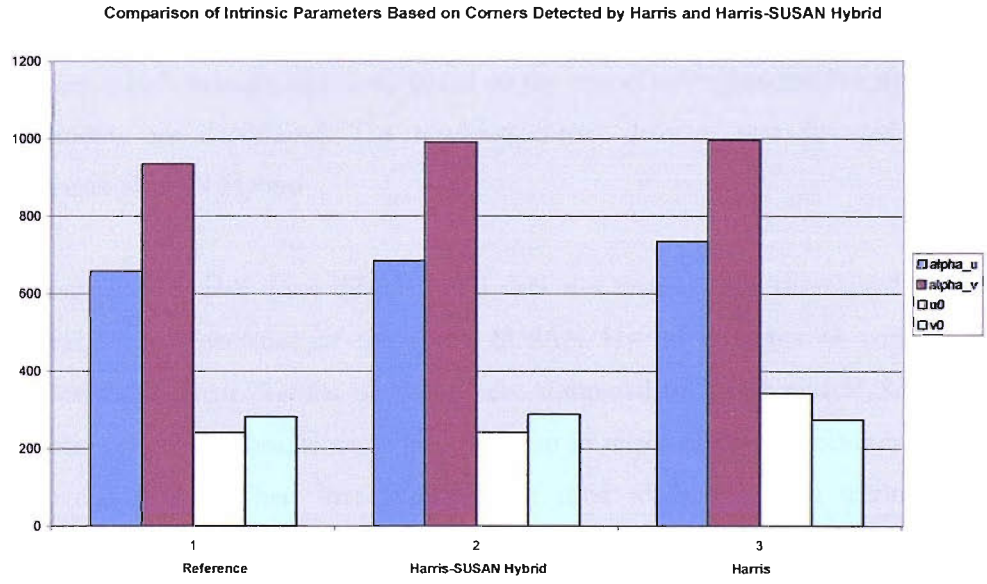


Figure 3.21: Comparison of intrinsic parameters obtained based on Harris and Harris-SUSAN Hybrid.

3.8 Conclusion

Feature detection is one of the important stages in AR system used to track the movement of marker or fiducials. It becomes more important when self-calibration of AR camera is to be included in the AR system. This is because the corners detected will be used for fundamental matrix estimation. In this chapter we propose corner detection to be the first pre-calibration stages for self-calibration in AR based on the fact that corners are the most available features in a scene.

We pointed out that based on [Sojka 2003], Harris corner detector outperforms most of available corner detectors in terms of higher number of correct matches and lower false matches. Based on these reasons, we have chosen Harris to be our preferred corner detector. We also pointed out that fundamental matrix estimation requires good localised corners for accurate estimation. However, Harris corner detector produces quite poor localised corners. Therefore, we proposed a new refinement step to improve the corner localisation. The proposed algorithm is based on calculating area inside a circular mask in search of true corner position. Different sizes of mask which changes adaptively based on the connected region inside a mask window are introduced. The resulting corner detector was denoted as Harris-SUSAN Hybrid.

Experiments with three different real data test images were performed to find the performance of the Harris-SUSAN Hybrid in terms of corner localisation error. Results obtained were compared to Harris and SUSAN corner detector. Results show improvement in terms of corner accuracy to several pixels. When these corners are used to find camera intrinsic parameters, the results shows some improvements compared to when Harris detected corners are used.

It is sometimes unavoidable to have some false corners, and false corners can be discarded during our point correspondence matching stage. In the self-calibration of AR it is very important to have a high accuracy of corners in order to minimise error propagation through the next stage, especially in the fundamental matrix estimation stage, which is very sensitive to corner location. This explains why corner accuracy is our main focus in this section and not merely the detection itself.

Sub pixel accuracy is one way to improve the algorithm in the future and is not included in this section due to time constraints.

Chapter 4

POINT CORRESPONDENCE MATCHING

4.1 Introduction

The main goal for point correspondence matching is to find the right pair of detected corners between two images. In this stage any corner that has no match is discarded leaving the same number of corners in both images. Even though we are dealing with a sequence of images, we have focused our implementation on solving the point correspondence matching problem between three captured frames from a sequence of images at different time instants. This is because in this stage the output that we want is not for tracking objects but as the input for the fundamental matrix estimation. Therefore, we will not be trying to solve the general problem of multi-frame point correspondence (e.g. Salari and Sethi [1990]), which is categorised as NP Hard for three or more frames. Matching in dynamic images (e.g. moving cars, clouds, etc.) is also not in our application area; instead, we will be focusing on finding matched points for a non-moving object or background.

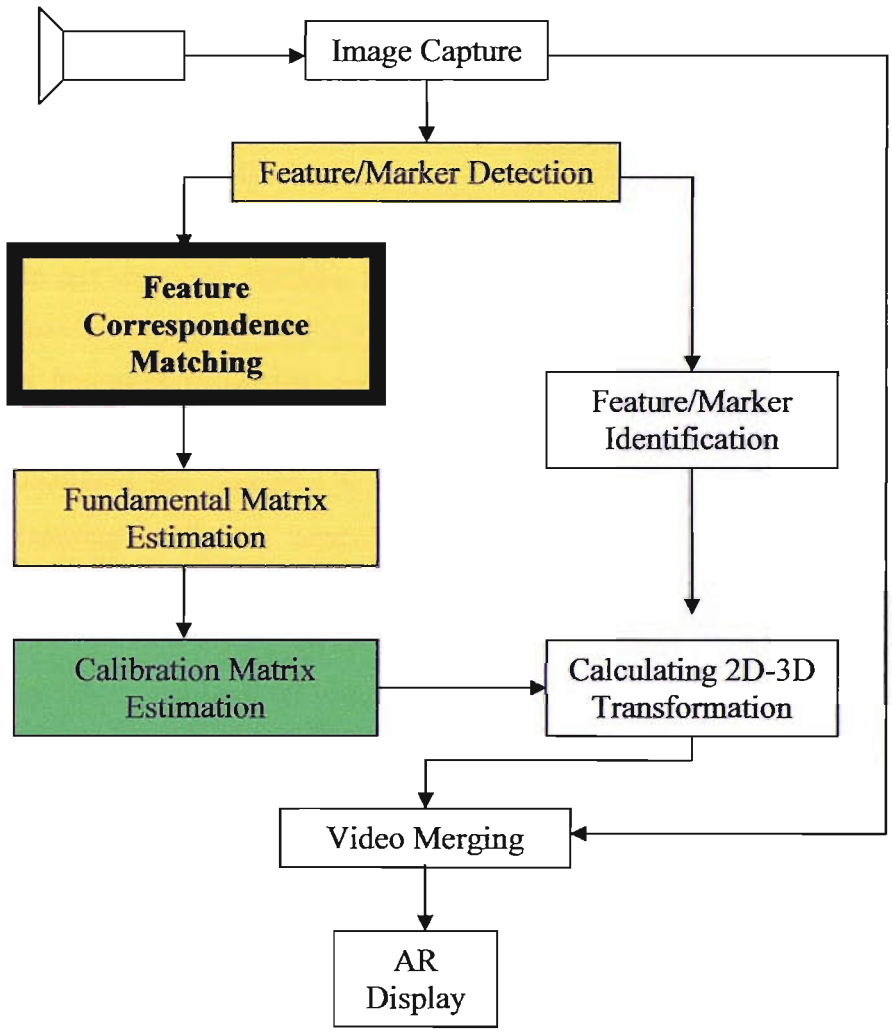


Figure 4.1: The position of the feature correspondence matching stage in the proposed self-calibration for AR system.

To apply self-calibration in AR that can update intrinsic parameters online requires the point correspondence stage to be efficient. Therefore, in this chapter we propose a novel way of matching point correspondence based on motion vector. We will show how our approach is different from other available techniques by exploiting motion vector and a simple statistical method. To achieve this, some matching techniques in the literature are discussed in the following section.

In our approach, matching by correlation is used to find an initial set of matches and we then use our new technique to discard false matches, which is important to provide correct and accurate input for the fundamental matrix estimation stage. Note that the accuracy of the input has been addressed and improved in the feature detection stage (Chapter 3) by reducing localisation error. The position of the point correspondence matching stage in the self-calibration for AR system is shown in Figure 4.1. We will discuss the suitability of the new algorithm to be applied to a sequence of images, and also prove that it can be used to tackle problems with occlusion during the matching process for an AR application. The results and discussion are provided at the end of the chapter.

4.2 Literature Review

In recent years, a large number of correspondence-matching algorithms have been proposed. One criterion that is most commonly used is the correlation of image pixels [Torr 1995; Lucas and Kanade 1981; Zhang et al. 1994], which is based on the assumption of image similarity.

Berthilsson and Astrom [1997] and Sudhir et al. [1997] solve correspondence based on assuming the rigidity of the 3D scene while Ohta and Kanade [1985] used the smoothness of the disparity field to solve the ambiguity between multiple solutions.

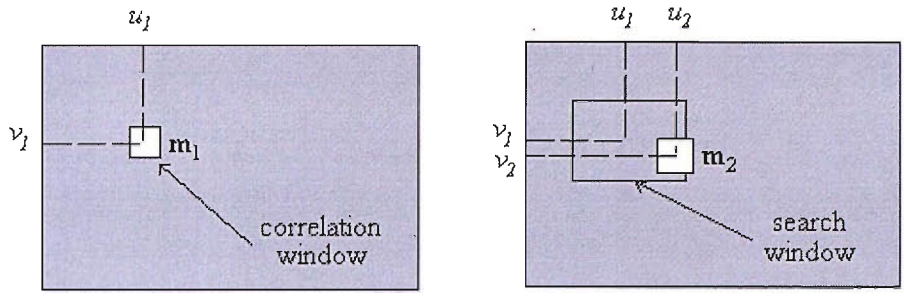
Other approaches used constraints, such as the epipolar constraint by Zhang et al. [1994], which proposed a robust approach to image matching by using classical technique (correlation and relaxation) to find an initial set of matches, and then used the Least Median of Squares (LMedS) to discard false matches. Epipolar geometry is then estimated and more matches are eventually found by using the recovered epipolar geometry. Other similar works using epipolar constraint include Ohta and Kanade [1985] and Roy and Cox [1997]. Besides epipolar, unicity constraint was also proposed by Gold et al. [1998].

The approach we propose in this chapter aims at exploiting motion vectors between two successive frames to establish correspondence between two perspective images of a single scene. We first detect corner points and then match them using correlation followed by a new technique based on motion vector to find the correct matches.

4.3 Correspondence Matching through Correlation

Matching detected corner points between two uncalibrated views can be achieved through a correlation-based matching algorithm. If corner detection is perfect, then each point in the first image will have its corresponding point in the second image.

In our implementation, a classical correlation technique is used to find initial matching candidates between two images. This is based on the assumption that the perspective changes between the successive frames are small.



(a) Image I_1 (b) Image I_2

Figure 4.2: Illustration of a correlation process.

Figure 4.2 illustrates the correlation process. $\mathbf{m}_1$ is the detected corner point, u_1 and v_1 are the pixel coordinates of $\mathbf{m}_1$ in horizontal and vertical directions respectively. The correlation window in Figure 4.2(a) is of size $(2n+1)\times(2m+1)$ centred at point $\mathbf{m}_1$ where n and m are the minimum distances in pixels between the centre point of the window and the points on the horizontal and vertical edges respectively.

The search window as shown in Figure 4.2(b) is of size $(2d_u+1)\times(2d_v+1)$ centred at coordinate (u_1, v_1) where d_u and d_v represent half of the width of the window size respectively. By using a search window, the search area for a corresponding point is reduced from the whole image to a given window. The correlation score is defined as Equation (4.1) where $\overline{I_k(u,v)} = \sum_{i=-n}^{n}\sum_{j=-m}^{m} I_k(u+i, v+j)/[(2n+1)(2m+1)]$ is the average at point (u,v) of image $I_k(k=1,2)$, and $\sigma(I_k)$ is the standard deviation of the image I_k in the neighbourhood $(2n+1)\times(2m+1)$ of (u,v), which is represented by Equation (4.2).

$$Score(\mathbf{m}_1, \mathbf{m}_2) = \frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m}\left[I_1(u_1+i, v_1+j) - \overline{I_1(u_1, v_1)}\right] \times \left[I_2(u_2+i, v_2+j) - \overline{I_2(u_2, v_2)}\right]}{(2n+1)(2m+1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}} \tag{4.1}$$

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m} I_k^2(u,v)}{(2n+1)(2m+1)} - \overline{I_k(u,v)}} \tag{4.2}$$

A score of -1 indicates two correlation windows that are not similar at all, whereas a score of 1 implies two correlation windows that are identical.

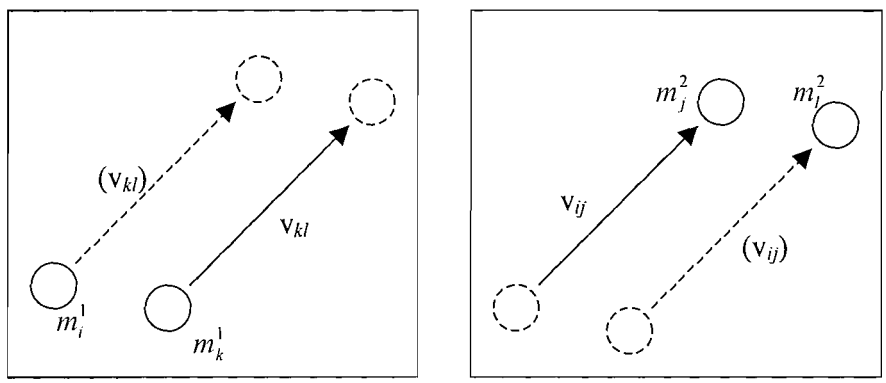
It is a common practice to apply a certain threshold in order to select the most probable matches. By doing this we have selected candidates for matches. The number of candidate matches will depend on how high the threshold is set. The higher the threshold the less the number of candidate matches we will get. The situation where the number of candidate matches is more than one is known as matching ambiguities.

Matching ambiguities occur when a point in the first image is paired to several points in the second image, namely the candidate matches. This occurs when the correlation technique using a certain threshold as discussed in this section is used. One of the widely used techniques for resolving matching ambiguities is known as the relaxation technique. In our implementation, to reduce the complexity of the algorithm the matches that have the highest correlation score will be selected as the most suitable matches. The problem with this approach is that the point found might not be the desired matched point, due to differences in image intensities between the two matches.

4.4 Determining Correct Matches through Motion Vector

Let m_i^1, m_k^1, m_j^2, m_l^2 be the feature points of two images. If m_i^1 and m_k^1 match m_j^2 and m_l^2 respectively, we can expect that the motion vector of m_i^1 to m_j^2 is approximately similar to that of m_k^1 to m_l^2. This idea is illustrated as in Figure 4.3.

If $\mathbf{v}_{pqx} = m_{qx}^2 - m_{px}^1$ and $\mathbf{v}_{pqy} = m_{qy}^2 - m_{py}^1$ where $\mathbf{v}_{pqx}$ represents a vector in x direction (from left to right of the image) and $\mathbf{v}_{pqy}$ represents a vector in y direction (from top to bottom of the image), then $\mathbf{v}_{ijx}$ should be similar to $\mathbf{v}_{klx}$ and $\mathbf{v}_{ijy}$ should be similar to $\mathbf{v}_{kly}$. These constraints can be used if we are absolutely certain as to which pair is the correct match. A pair that has a correct match can be the *reference vector* in order to determine other matching pairs.



(a) Image I_1 (b) Image I_2

Figure 4.3: Similarity measure of relative vectors for two matched feature point pairs.

If m_i^1 and m_j^2 are really the correct matching points, naturally we will have many vectors that are similar to $\mathbf{v}_{ij}$ plus some outliers. To find the *reference vector* we do the following:

(a) We perform correlation and choose any pair that has the maximum correlation score to be candidate matches.

(b) Let $\mathbf{M}_{xi}^1$ and $\mathbf{M}_{yi}^1$ be arrays of x-coordinates and y-coordinates of feature points in image I_1 respectively that is $i = 1 \ldots n$ where n is the total number of matches.

(c) Let $\mathbf{M}_{xi}^2$ and $\mathbf{M}_{yi}^2$ be arrays of x-coordinates and y-coordinates of corresponding matches in image I_2 respectively, that is $i = 1 \ldots n$ where n is the total number of matches.

(d) We define $\mathbf{V}_{xi} = \mathbf{M}_{xi}^2 - \mathbf{M}_{xi}^1$ and $\mathbf{V}_{yi} = \mathbf{M}_{yi}^2 - \mathbf{M}_{yi}^1$ as arrays of vectors in x direction and y direction.

(e) We choose the highest occurrence of $\mathbf{V}_{xi}$ that is $\text{mode}(\mathbf{V}_{xi})$ and highest occurrence of $\mathbf{V}_{yi}$ that is $\text{mode}(\mathbf{V}_{yi})$ to be our *reference vectors*.

(f) Let $\mathbf{D}_{vxi}$ be arrays of the difference between $\mathbf{V}_{xi}$ and its mode and $\mathbf{D}_{vyi}$ be arrays of the difference between $\mathbf{V}_{yi}$ and its mode, that is $\mathbf{D}_{vxi} = \mathbf{V}_{xi} - \text{mode}(\mathbf{V}_{xi})$ and $\mathbf{D}_{vyi} = \mathbf{V}_{yi} - \text{mode}(\mathbf{V}_{yi})$.

(g) To determine whether the i^{th} match is a correct match, $\mathbf{D}_{vxi}$ and $\mathbf{D}_{vyi}$ must fulfill constraint $|D_{vxi}| \bullet |D_{vyi}| \leq R$ where R is the radius of difference in pixels defined by the user. This number can be between 1 and 10.

(h) If the above constraint is fulfilled, consider the i^{th} pair as a good match, otherwise discard it.

4.5 Experimental Setup

We are using *big_house_frame sequence* from website http://www.cv.iit.nrc.ca/~gerhard/PVT/. The sequences were chosen to show the ability of our algorithm to find point correspondence successfully even though there are many repetitive patterns. Repetitive patterns may occur in an AR scene where 2 or more markers are present. During implementation, the points were extracted using the Harris corner detector as a standard method for comparison with other point correspondence technique. The values of n and m for the correlation window are set to 7. For the search window, the values of d_v and d_u are set to a quarter of the image height and width, respectively. This equals half of the image area. The radius R is set to 7 and the threshold for the correlation score is set to 0.8. These values are chosen as they give the best results empirically. We compare our results with the famous and established Image Matching Software developed by Zhang [1994].

4.6 Results and Discussion

Figure 4.8 shows the comparison in terms of the total number of correct matches between our algorithm and Zhang's algorithm for house sequence. Our algorithm proves to outperform Zhang's algorithm, as the percentage of correct matches from our algorithm varies between 95 and 99 percent whereas Zhang's algorithm varies between 89 and 96 percent.

Figure 4.4 and 4.5 show the matches found by our algorithm and Zhang's algorithm respectively when the camera undergoes general motion. Based on observation, our matched points show 98% correct matches, whereas Zhang's algorithm shows 96% correct matches. An example of incorrect matches found by our algorithm is shown by match number 1 in Figure 4.4, which is not severe compared with matches' number 14, 18 and 17 in Figure

4.5 by Zhang's algorithm. Figures 4.6 and 4.7 show the matches found when the camera undergoes forward motion. Both algorithms perform well, but the incorrect matches found by Zhang's algorithm (matches number 25 and 29 in Figure 4.7) are more severe than the incorrect matches found by our algorithm (matches number 7 and 9 in Figure 4.6).

Since matching is based on motion vector, if any of the matched pairs are wrongly matched due to occlusion (refer Figure 4.8 and Figure 4.9), the algorithm will detect it simply by the *reference vector*. Our algorithm can be used effectively to find correct matches for repeated texture. As long as the number of correct matches is more than 50% of the whole data, this algorithm will produce a nearly perfect ratio of correct matches.

One of the weaknesses of this algorithm is that it is not suitable for two images that are widely separated from each other. However, as the images are from a sequence of frames, the difference in perspective views is not too apparent, enabling our algorithm to suit the AR application.

Figure 4.9 and Figure 4.10 illustrate the comparison between Zhang's and our algorithm in terms of the number of correct matches when there is occlusion in the scene. In this image pair, the occlusion objects are the finger and the laptop. Figure 4.9 shows matches from Zhang's algorithm when there is an occlusion, which gives 88% correct matches. Figure 4.10 shows our algorithm, which gives 98% correct matches. From Figure 4.9, it is shown that corners number 126, 132, 133, 143, and 165 are poorly matched due to the occlusion caused by the laptop. Similarly corners number 73, 125, and 134 and 125 are also poorly matched due to the occlusion caused by the finger. On the other hand, referring to Figure 4.10, our algorithm performs better in the case of occlusion. None of the corner points are affected by both occluding objects. Hence, the number of correct matches, which is 98%, is better than Zhang's, which attains 88%.

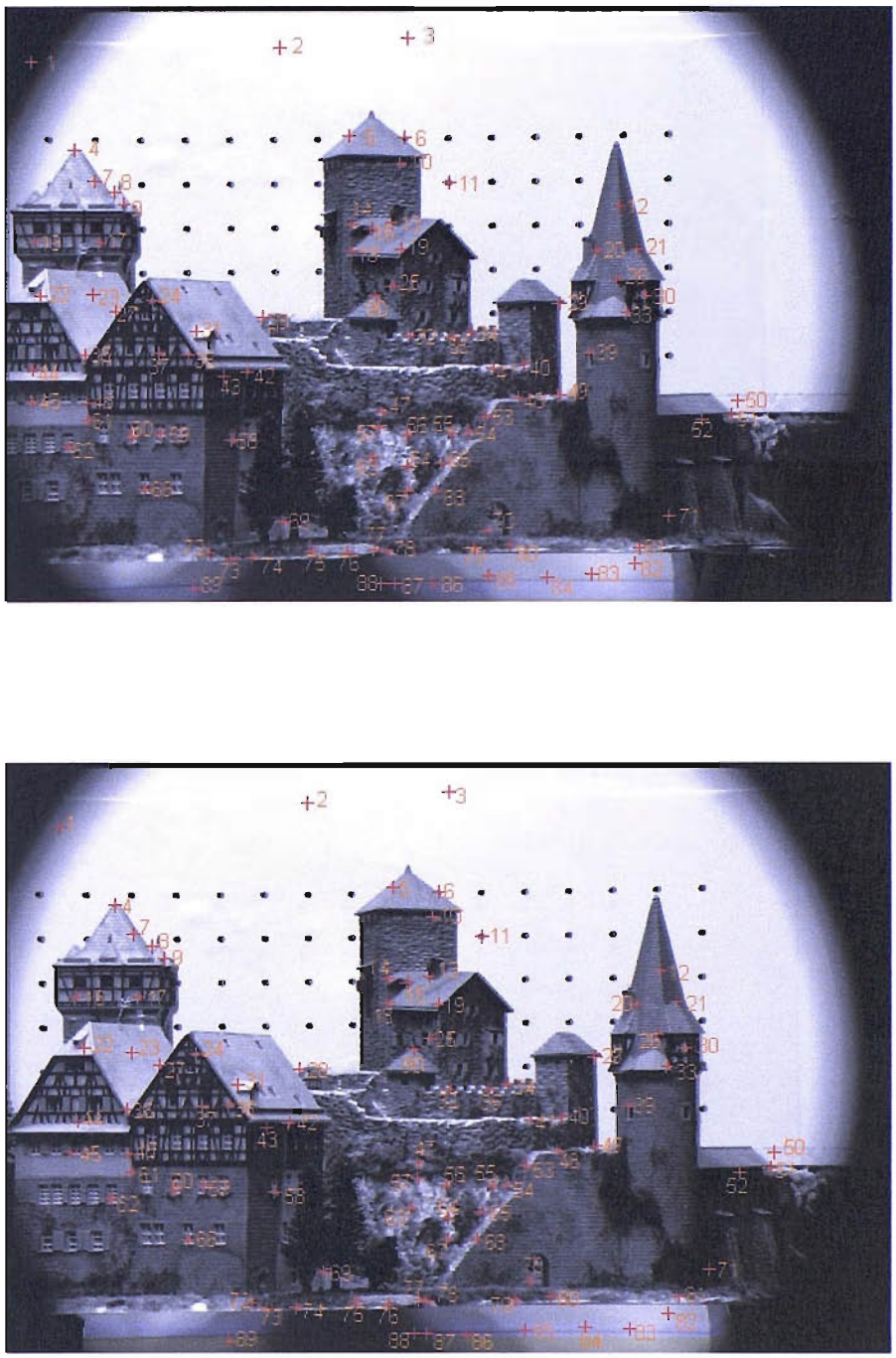


Figure 4.4: Feature point matching between first and second frame. There are 89 pairs of feature points. The ratio of correct matches is 98% using our algorithm.

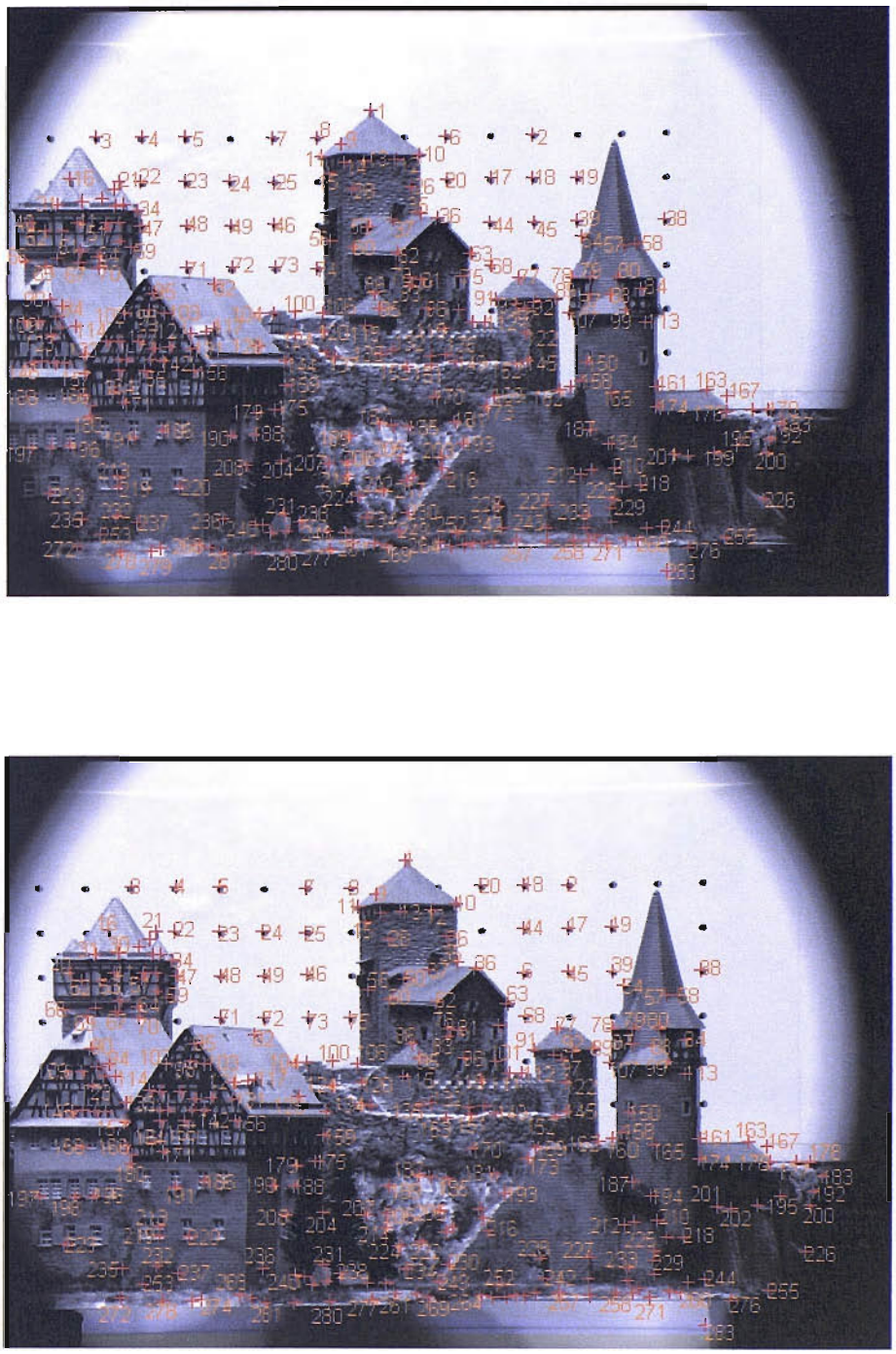


Figure 4.5: Feature point matching between first and second frame. There are 283 pairs of feature points. The ratio of correct matches is 96% using Zhang's.

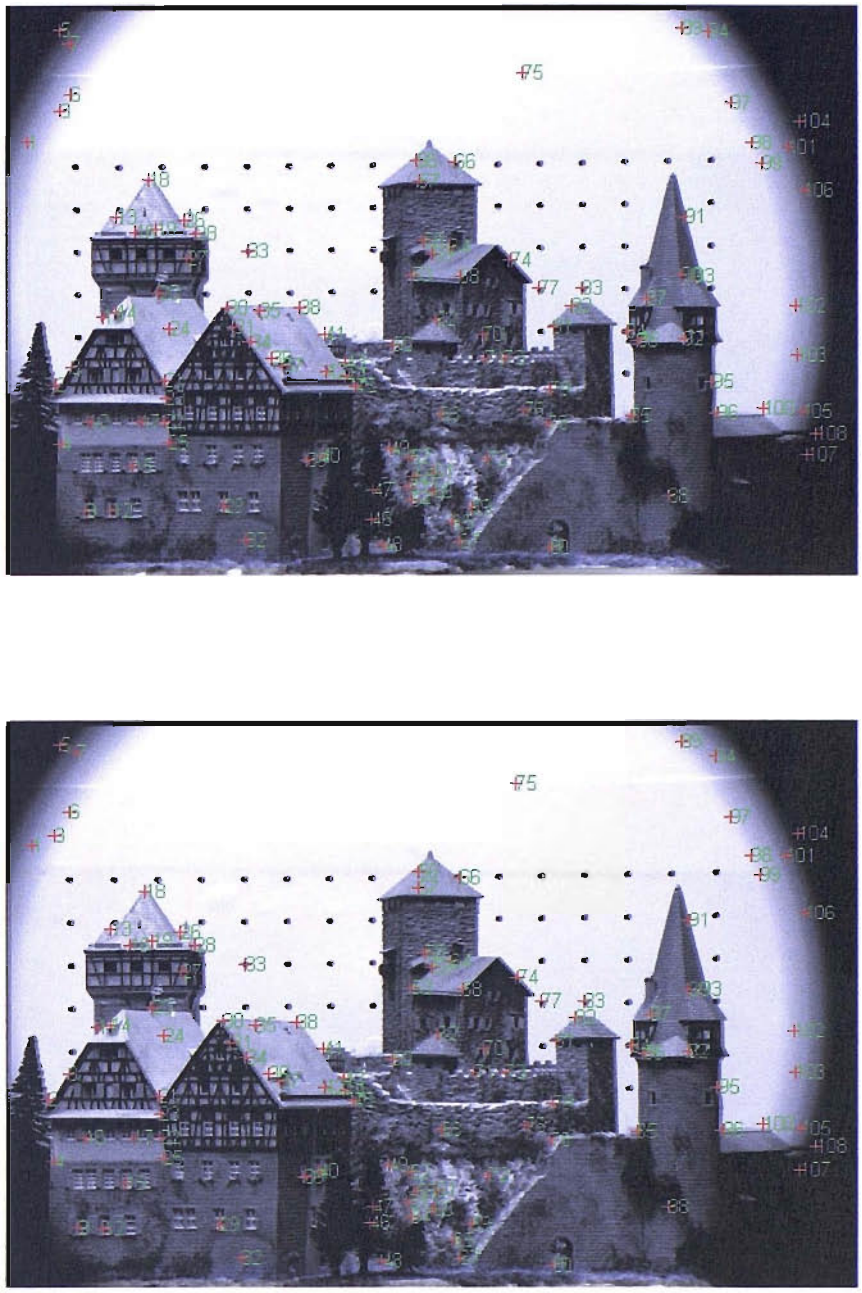


Figure 4.6: Feature point matching between 7th and 9th frame under forward motion. There are 108 pairs of feature points. Ratio of correct matches is 98% using our algorithm.

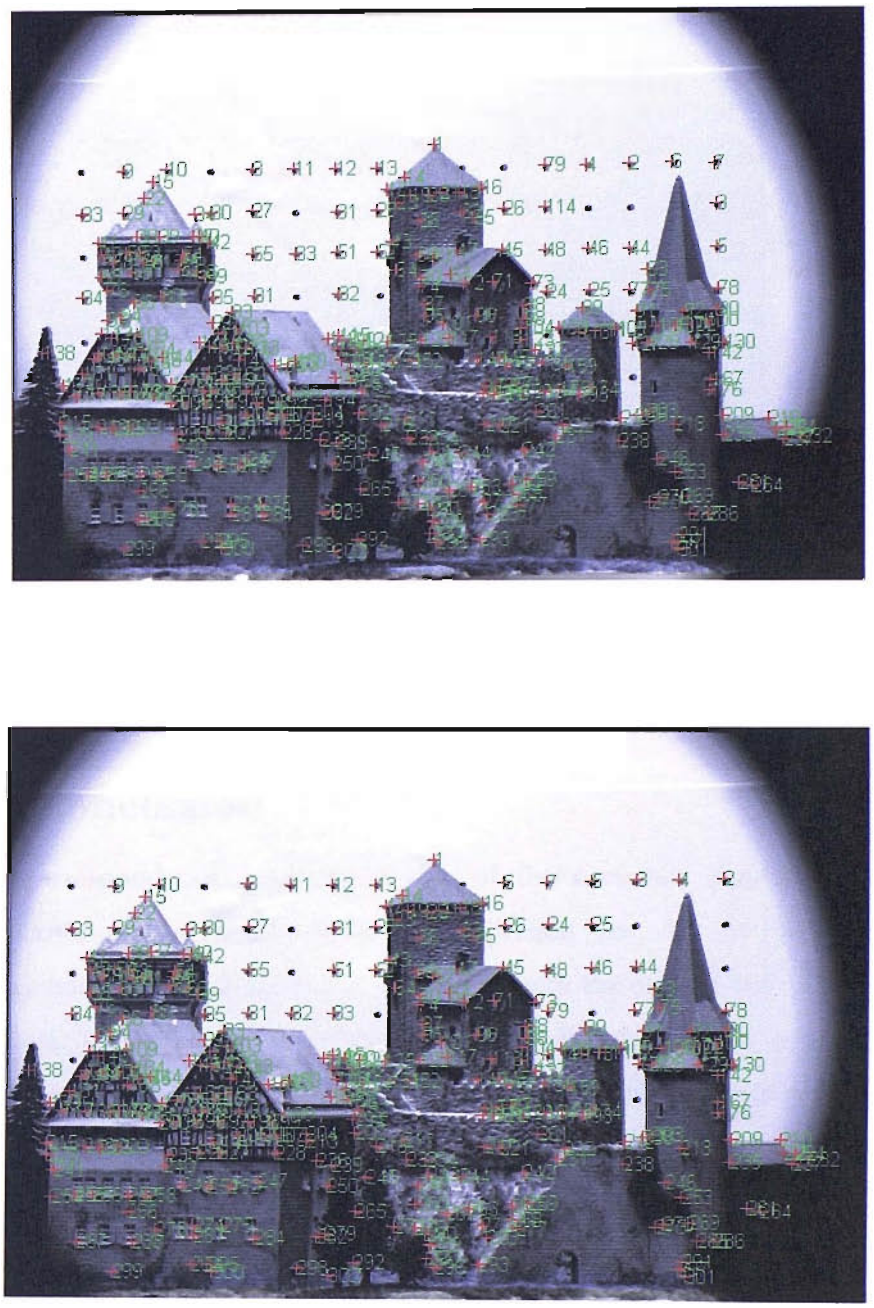


Figure 4.7: Feature point matching between 7th and 9th frame with forward motion. There are 302 pairs of feature points. Correct matches ratio is 86% using Zhang's.

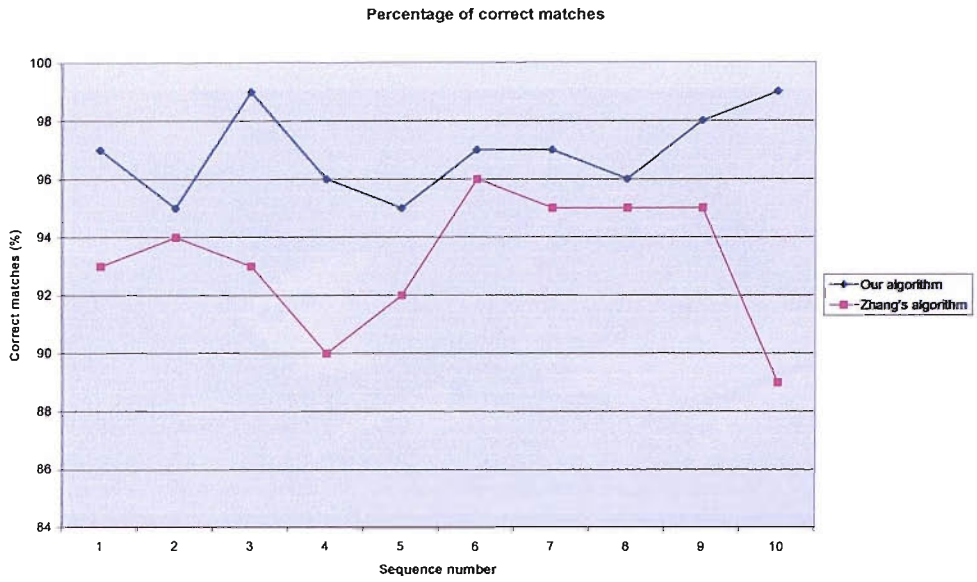


Figure 4.8: Comparison number of correct matches between our algorithm and Zhang's algorithm for house sequence.

4.7 Conclusion

Point correspondence matching is one of the necessary pre-calibration stages after corner detection in self-calibration for AR. It finds the correspondence of each corner in first image with the second one which is important as the input for the next pre-calibration stage which is fundamental estimation stage. Correct corner matches ensures good estimation of the fundamental matrix.

In this chapter, we proposed a simple and robust algorithm for point correspondence matching as one of the pre-calibration stages for an AR system. The algorithm is based on simple correlation technique and motion vector between two images separated by general motion. We introduced *reference vectors* (in horizontal and vertical direction) as the measures of how close a motion vector of a point to the true vector. This is determined from the number of highest occurrence vector in horizontal and vertical directions.

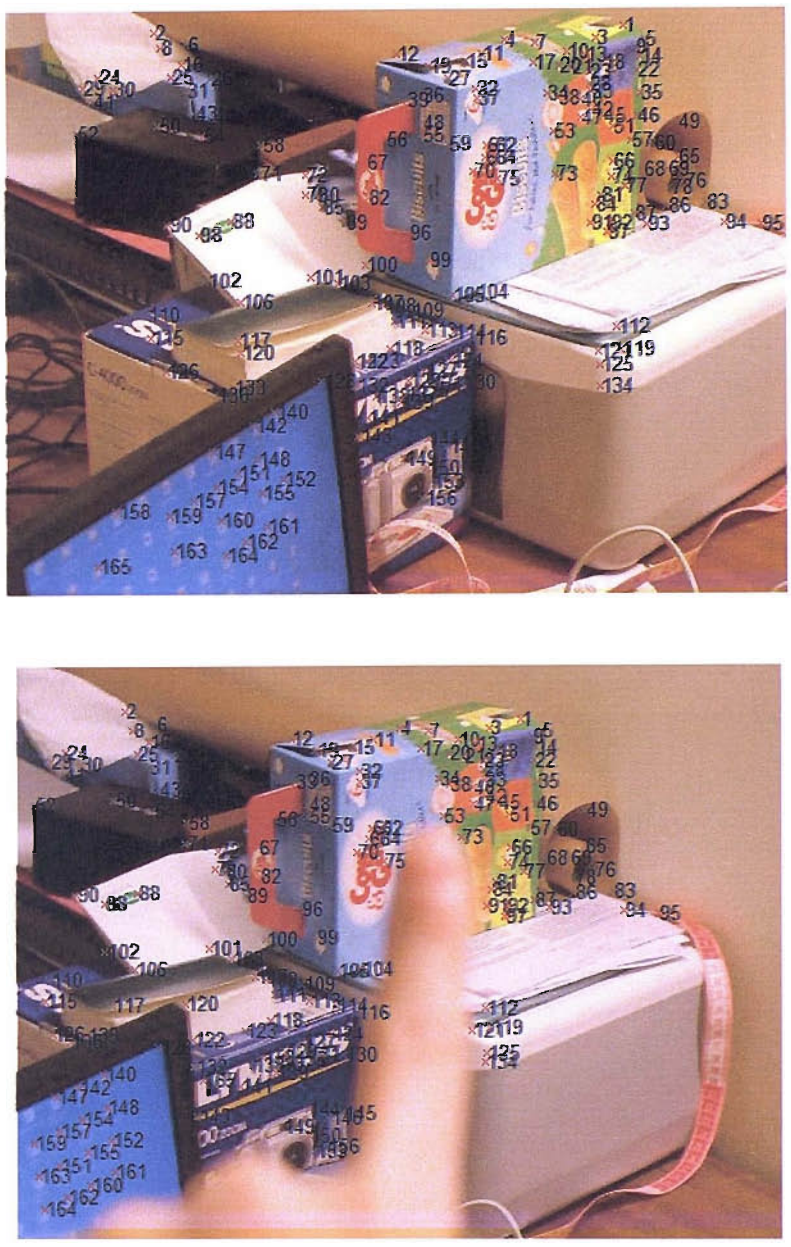


Figure 4.9: Matches from Zhang's algorithm when there is an occlusion showing 88% correct matches.

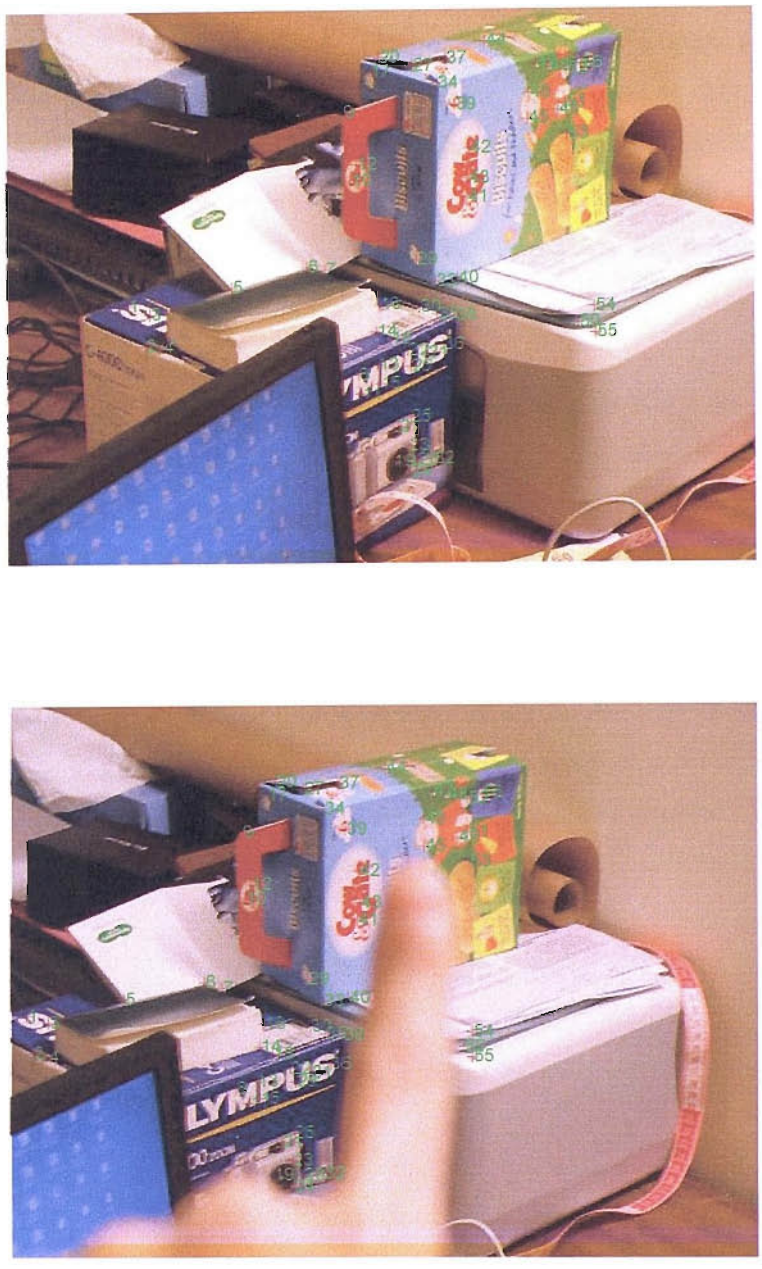


Figure 4.10: Our algorithm showing 98% correct matches.

Experimental results show that by using our algorithm we can discard outliers effectively for house sequence images even though there are repetitive patterns in the scenes. This also applies when the camera is moving forward or backward. The performance is always superior compared with algorithm proposed by Zhang [1994]. As it chooses the points with the maximum correlation as the most probable match, it only needs to scan the image once and there is no need for any further relaxation process. This shows its efficiency, which is important for real-time AR applications.

Apart from that, our algorithm shows better performance when the scene is occluded whether intentionally or unintentionally. The number of correct matches is still better in comparison to Zhang's. One disadvantage of our proposed algorithm is that the number of matches found is as not many as the one found by Zhang. This is due to the fact that we only select the points that have the highest correlation score to be our matches. This approach is less effective for a scene, which contains pure repetitive structure. Possible improvement might include finding new matches based on the reference vector especially for a scene with repetitive object. This will however increase the complexity of the algorithm.

Chapter 5

CAMERA SELF-CALIBRATION BASED ON THREE VIEWS

5.1 Introduction

This chapter addresses the problem of self-calibrating a camera based on three captured views taken from a sequence of frames and based on algebraic constraints derived by Dornaika and Chung [2001]. Its main contribution to AR research is the use of an algebraic approach in the development of self-calibration for AR system. This is incorporated with a technique that simultaneously recovers the fundamental matrix and distortion parameters which provides the input for calibration matrix estimation stage. Only three captured views are needed from the camera input, although the proposed method can equally be used with more than three views. In our system, these three views/frames need to undergo corner detection, point correspondence matching and fundamental matrix estimation before they can be used to find the camera parameters. Figure 5.1 illustrates the location of the self-calibration process in the system. The input involved is solely from the fundamental matrix and the output is camera intrinsic parameters, which are then used in the subsequent process for calculating the 2D-3D transformation matrix.

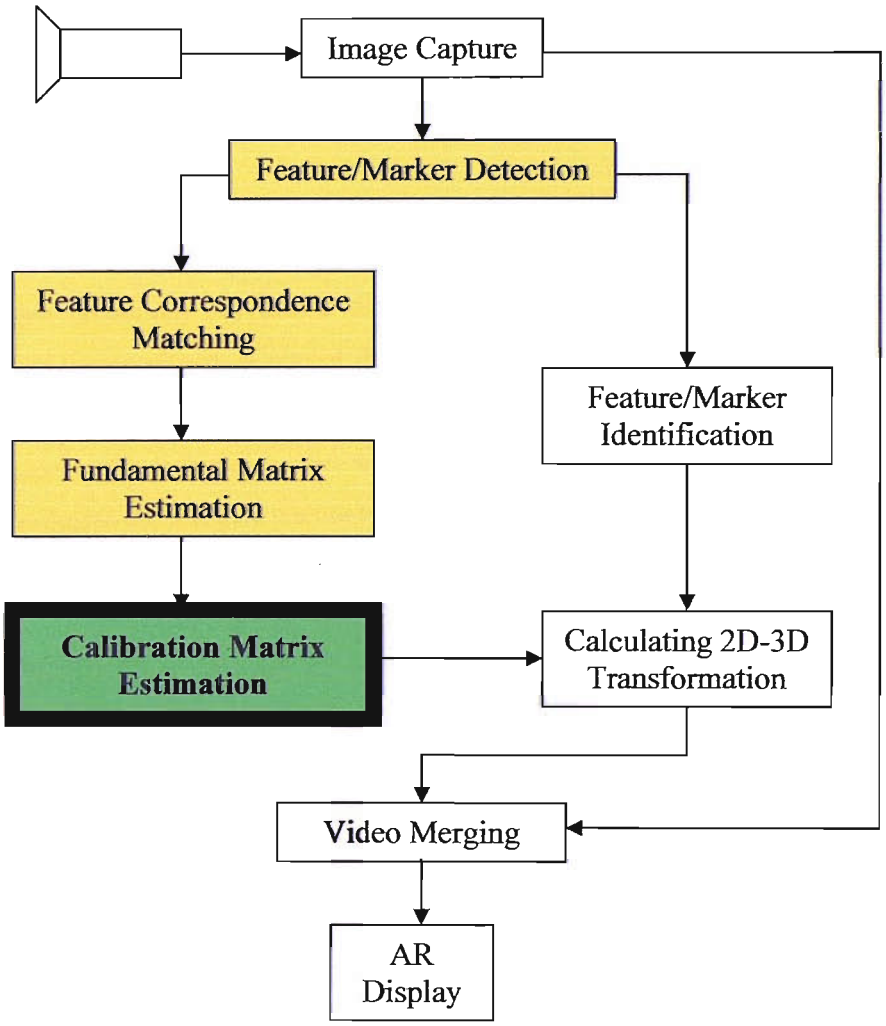


Figure 5.1: The position of Calibration Matrix Estimation (Self-Calibration based on three views) in the proposed self-calibration for AR system.

Also this chapter contains details of the method of derivation of the algebraic constraints and of how the distortion parameter estimation is integrated into the self-calibration. Experiments with synthetic and real data have shown that the technique can estimate camera intrinsic parameter accurately and is quite reliable for highly distorted images.

5.2 Literature Review

Self-calibration has drawn the attention of researchers in the computer vision community as one of the effective methods of recovering 3D models from sequences of images. It is different from normal classical calibration due to the fact that it does not require any particular structure in order to estimate the intrinsic parameters. Instead, it uses constraints that are derived from the projective and epipolar geometry of the sequence of images.

Some of these constraints are expressed as the Trivedi constraints [Trivedi 1988], the Huang and Faugeras constraints [Hartley 1992; Mendonca and Cipolla 1999], and the Kruppa equations [Lourakis and Deriche 2000; Luong and Faugeras 1997]; are formulated in terms of absolute quadric [Pollefey et al. 1999; Triggs 1996] and also algebraic constraints [Dornaika and Chung 2001; Abdullah and Martinez 2002].

Perhaps the earliest introduction to the concept of self-calibration in computer vision was done by Maybank and Faugeras [1992], who established the relationship between intrinsic parameters and absolute conic, from which were then developed the Kruppa equations. Hartley [1992] used Huang and Faugeras constraints to develop a self-calibration algorithm that estimates the focal lengths of two cameras based on a known corresponding fundamental matrix and other intrinsic parameters. Hartley [1994a] then developed an algorithm for self-calibration for more than three cameras based on the method proposed by Maybank and Faugeras [1992]. He also

introduced the idea of updating a projective reconstruction to a Euclidean one through 3D homography. Based on this approach, Triggs [1997] introduced absolute quadric while Pollefeys et al. [1998] developed the self-calibration of multiple cameras with varying and unknown intrinsic parameters.

In this chapter, we employ the constraints derived by Dornaika and Chung [2001] to find camera intrinsic parameters based on three views for an AR system. In our implementation we assume that shear measure $b = 0$ and that other intrinsic parameters are unknown and can be varied throughout sequence of images. The algorithms developed will also takes into account distortion correction should the image captured be heavily distorted The method takes only the fundamental matrix which has been estimated from the previous stage (refer Figure 5.1) as an input to find the intrinsic parameters.

Note that the third pre-calibration stages which is the fundamental matrix stage, is not specified into one chapter. This is because most of the theories and literature reviews have been covered in Chapter 2. This does not imply however that this stage is less significant but it is just as important as other pre-calibration stages. In our implementation we propose the use of MAPSAC algorithm described in [Torr 2002] in order to estimate the fundamental matrices. The reader can refer to Chapter 2 for more details on how fundamental matrices are derived and how it can be found from point correspondences. Different techniques employed for fundamental matrix estimation also can be found in Chapter 2.

5.3 Derivation of the Method

Recall from Chapter 2 that for a single camera undergoing general motions, the fundamental matrix equation is $\mathbf{F} = (\mathbf{K}^{-1})^T \mathbf{S(t)R}(\mathbf{K}')^{-1}$. This equation can be written as

$$\mathbf{K}^T\mathbf{F}\mathbf{K} \cong \mathbf{S(t)R} \tag{5.1}$$

where we assume that $\mathbf{K} = \mathbf{K}'$ (same camera is used for two views or both cameras have the same calibration matrix). Multiplying both sides with $\mathbf{R}^T$,

$$\mathbf{K}^T\mathbf{F}\mathbf{K}\mathbf{R}^T \cong \mathbf{S(t)}. \tag{5.2}$$

Recall that $\mathbf{S(t)}$ is a skew-symmetric matrix, hence if fundamental matrix $\mathbf{F}$ is known, we can obtain six polynomial constraints on the entries of $\mathbf{K}$ and $\mathbf{R}$. Let a_{jk} where $1 \le j,k \le 3$ be the entries of $\mathbf{K}^T\mathbf{F}\mathbf{K}\mathbf{R}^T$, then

$$a_{11} = a_{22} = a_{33} = a_{12} + a_{21} = a_{13} + a_{31} = a_{23} + a_{32} = 0 \tag{5.3}$$

Let f_{jk} and r_{jk} be the entries of the matrices $\mathbf{F}$ and $\mathbf{R}$ respectively where $1 \le j,k \le 3$. Equation (5.3) can be written as:

$$f_{11}r_{11}\alpha_u^2 + (f_{12}r_{12}\alpha_v + (f_{11}u_0 + f_{12}v_0 + f_{13})r_{13})\alpha_u = 0 \tag{5.4}$$

$$f_{22}r_{22}\alpha_v^2 + (f_{21}r_{21}\alpha_u + (f_{21}u_0 + f_{22}v_0 + f_{23})r_{23})\alpha_v = 0 \tag{5.5}$$

$$\begin{aligned}(f_{11}u_0 + f_{21}v_0 + f_{31})\alpha_u r_{31} + (f_{12}u_0 + f_{22}v_0 + f_{32})\alpha_v r_{32} + ((f_{11}u_0 + f_{21}v_0 \\ + f_{31})u_0 + (f_{12}u_0 + f_{22}v_0 + f_{32})v_0 + f_{13}u_0 + f_{23}v_0 + f_{33})r_{33} = 0\end{aligned} \tag{5.6}$$

$$f_{11}\alpha_u^2 r_{21} + f_{12}\alpha_u\alpha_v r_{22} + (f_{11}\alpha_u u_0 + f_{12}\alpha_u v_0 + f_{13}\alpha_u)r_{23} + f_{21}\alpha_v\alpha_u r_{11} + f_{22}\alpha_v^2 r_{12} + (f_{21}\alpha_v u_0 + f_{22}\alpha_v v_0 + f_{23}\alpha_v)r_{13} = 0 \tag{5.7}$$

$$f_{11}\alpha_u^2 r_{31} + f_{12}\alpha_u\alpha_v r_{32} + (f_{11}\alpha_u u_0 + f_{12}\alpha_u v_0 + f_{13}\alpha_u)r_{33} + (f_{11}u_0 + f_{21}v_0 + f_{31})\alpha_u r_{11} + (f_{12}u_0 + f_{22}v_0 + f_{32})\alpha_v r_{12} + ((f_{11}u_0 + f_{21}v_0 + f_{31})u_0 + (f_{12}u_0 + f_{22}v_0 + f_{32})v_0 + f_{13}u_0 + f_{23}v_0 + f_{33})r_{13} = 0 \tag{5.8}$$

$$f_{21}\alpha_v\alpha_u r_{31} + f_{22}\alpha_v^2 r_{32} + (f_{21}\alpha_v u_0 + f_{22}\alpha_v v_0 + f_{23}\alpha_v)r_{33} + (f_{11}u_0 + f_{21}v_0 + f_{31})\alpha_u r_{21} + (f_{12}u_0 + f_{22}v_0 + f_{32})\alpha_v r_{22} + ((f_{11}u_0 + f_{21}v_0 + f_{31})u_0 + (f_{12}u_0 + f_{22}v_0 + f_{32})v_0 + f_{13}u_0 + f_{23}v_0 + f_{33})r_{23} = 0. \tag{5.9}$$

where α_u and α_v are the horizontal and vertical scale factors respectively while u_0 and v_0 represent the x and y co-ordinates of the principal point.

The rotation $\mathbf{R}$ is represented by its associated unit quaternion $\mathbf{q} = (q_0, q_x, q_y, q_z)^T$. Therefore,

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{5.10}$$

The 6 constraints (Equation (5.4) to Equation (5.9)) can be written in a compact form of vector $\mathbf{v}$:

$$\mathbf{v} = 0, \tag{5.11}$$

where

$$\mathbf{v} = (a_{11},\ a_{22},\ a_{33},\ a_{12}+a_{21},\ a_{13}+a_{31},\ a_{23}+a_{32})^T \tag{5.12}$$

Note that these 6 constraints are associated with a pair of views only (one motion). We require at least three views (two pairs of views) before we can recover the intrinsic parameter $\mathbf{K}$ and the rotation matrix $\mathbf{R}$.

5.4 Algorithm

Let n represents the number of camera motions. The constraints (Equation (5.4) to Equation (5.9)) can be cumulated by building a positive error function f that will be minimised over the unknowns. Since the fundamental matrix can be obtained from point correspondence matching (refer Chapter 4) and epipolar geometry (refer Chapter 2), the unknowns will be the intrinsic parameters $\mathbf{K}$ and the quaternion $\mathbf{q}$. Therefore, the error function can be written as follows:

$$f(\alpha_u, \alpha_v, u_0, v_0, \mathbf{q}_1, \ldots, \mathbf{q}_n) = \sum_{i=1}^{n} [\| \mathbf{v}_i \|^2 + \lambda(1 - \| \mathbf{q}_i \|^2)^2] \tag{5.13}$$

where the second term in f is a penalty function that constrains the quaternion $\mathbf{q}_i$ (for camera motion i) to be a unit quaternion, and λ is a real positive number. Each $\mathbf{v}_i$ contains six polynomials where the coefficients are given by the corresponding fundamental matrix $\mathbf{F}_i$.

Since the error function f is the sum of positive functions, it can be written in such a way that it becomes a classical non-linear least squares constrained minimisation problem:

$$f(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{7n}\phi_j^2 \tag{5.14}$$

where $\mathbf{x} = (\alpha_u, \alpha_v, u_0, v_0, \mathbf{q}_1^T, \ldots, \mathbf{q}_n^T)^T \in \Re^{4+4n}$ for a four-parameter camera model and n is the number of camera motions. For each motion, the first six ϕ_j are given by Equation (5.4) to Equation (5.9). The implementation of the algorithm has been done using the Levenberg-Marquardt technique [Marquardt 1963] for its robustness.

We can simultaneously recover the intrinsic parameters and the rotation matrices by minimising the function $f(\mathbf{x})$ over the vector $\mathbf{x}$. For n motions, we have $4+4n$ unknowns and $7n$ constraints. Thus, for a calibration matrix with four parameters we need at least two motions so that we have 12 unknowns and 14 constraints, from which we are able to obtain a solution.

The weight λ can be set through the following:

- An arbitrary value is chosen and the non-linear algorithm is run until convergence is obtained.
- Then, the penalty functions $(1-\| \mathbf{q}_i \|^2)^2$ are evaluated for all motions.
- The weight has to be increased accordingly if the penalty functions are not close to zero.

5.4.1 Initialisation Procedure

The unknowns in the function $f(\mathbf{x})$ contain two components. The first component is the intrinsic parameters $(\alpha_u, \alpha_v, u_0, v_0)$ and the second component is the quaternion $\mathbf{q}_i = (q_{0i}, q_{xi}, q_{yi}, q_{zi})$. The intrinsic

parameters are initialised by using an educated guess or values provided by the manufacturers.

The initial estimate of the quaternion can be obtained as follows:

- From the initial estimate of the intrinsic parameter, the upper triangular matrix $\mathbf{K}$ is formed.

- Then the essential matrix $\mathbf{E}$ is calculated for each motion i using the relation:

$$\mathbf{E}_i \cong \mathbf{K}^T \mathbf{F}_i \mathbf{K} . \tag{5.15}$$

- Each quaternion $\mathbf{q}_i$ is then estimated by factorising the corresponding essential matrix as in Luong [1997]:

$$\mathbf{E}_i \cong \mathbf{S}(\mathbf{t}_i)\mathbf{R}(\mathbf{q}_i) \tag{5.16}$$

Note that in this algorithm the derivation of the 6 constraints was quite simple and purely algebraic. Note also that the initialisation of the intrinsic parameters is only done by an educated guess and does not require a good approximation to the actual value, making this algorithm more robust than other available self-calibration techniques for AR systems.

5.5 Dealing with Critical Motion

It is well known in several works that self-calibration of all parameters may not be possible for certain motions [Zeller and Faugeras 1996; Sturm 1997; Sturm 2002]. This kind of motion is called critical motion. Any attempt to self-calibrate a camera under critical motion will result in incorrect values for

intrinsic parameters. Such ambiguities may be resolved by adding further constraints, e.g. zero skew, square pixels or knowledge of the principal point. Nevertheless, even with these constraints applied, certain motions remain ambiguous. This fact is supported by work of Zisserman et al. [1998].

According to Sturm [2002], there are 3 types of critical motion sequences for a moving camera that can lead to failure in self-calibration:

1) Arbitrary position of optical centres but parallel optical axes. In other word, camera motions are pure translations possibly combined with an arbitrary rotation about the optical axis.

2) Collinear optical centres, which means that camera motions are pure forward translations with two exceptions where the translation may be followed by an arbitrary rotation about the optical centre.

3) The optical centres lie on an ellipse/hyperbola pair as shown in Figure 5.2. The views may be partitioned into at most two sets, for which the centres and optical axes are all coplanar.

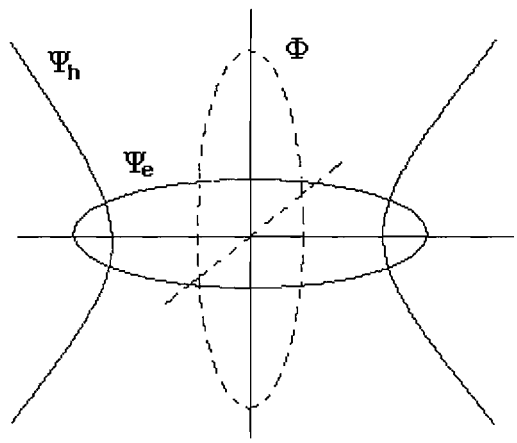


Figure 5.2: Locus of camera positions in a critical motion sequence with respect to a conic Φ not on the ideal plane. The conic Φ is shown in dotted style to illustrate that it is virtual and can in fact not be drawn. Refer to [Sturm 2002] for details.

In this section, a novel method has been developed to calculate how near the particular motion is to the critical motion associated with case 1 described above, based on the fact that the cross product of two parallel vectors equals zero.

Recall that the fundamental matrix equation can be written as $\mathbf{F} = \mathbf{K}^{-T}\mathbf{S(t)}\mathbf{R}\mathbf{K}^{-1}$ where $\mathbf{E} = \mathbf{S(t)}\mathbf{R}$. Given that the fundamental matrix has been estimated using SVD (that is from Equations (2.38) and (2.39)), we can estimate the rotation matrix $\mathbf{R}$ and skew-symmetric matrix $\mathbf{S(t)}$ between two views.

Let the first view of the motion correspond to a camera pointing in z direction, which means that the optical axis is equal to the z-axis. Let us assume that the vector pointing towards the z direction for the first view is

$$\mathbf{V}_1 = [0 \quad 0 \quad 1] \tag{5.17}$$

and the rotation matrix $\mathbf{R}$ is defined by equation (2.12). To find the vector for the second view $\mathbf{V}_2$ with respect to the first view we need to multiply $\mathbf{R}$ with $\mathbf{V}_1$:

$$\mathbf{V}_2 = \mathbf{R}\mathbf{V}_1^T \tag{5.18}$$

which gives

$$\mathbf{V}_2 = \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \tag{5.19}$$

If $\mathbf{V}_1$ is crossed product with $\mathbf{V}_2$, equation (5.19) becomes:

$$\mathbf{V}_3 = \begin{bmatrix} r_{23} \\ -r_{13} \\ 0 \end{bmatrix} \tag{5.20}$$

Let the translation $\mathbf{t} = 0$, then the angle Ψ between vector $\mathbf{V}_1$ and $\mathbf{V}_2$ is given by:

$$\Psi = \tan^{-1}\left(\frac{r_{23}}{r_{13}}\right) \tag{5.20}$$

The magnitude of vector $\mathbf{W}$ is given by:

$$w = |\mathbf{V}_3| = \sqrt{r_{13}^2 + r_{23}^2} \tag{5.21}$$

where w is the measure of the closeness of the motion to being critical.

Figure 5.3 illustrates the value of w plotted for different elevation angles τ and vergence angles v. It shows the usefulness of w in Equation (5.21) as a measure of closeness to case 1 critical configuration between the motions of a given pair of cameras. The bigger the value of w, the less likely that the pair of views is involved in critical motion.

In our implementation, we set the threshold $w = 0.1$ to determine whether a motion is involved in critical motion. If yes, then the pair will be discarded and changed for another pair of views.

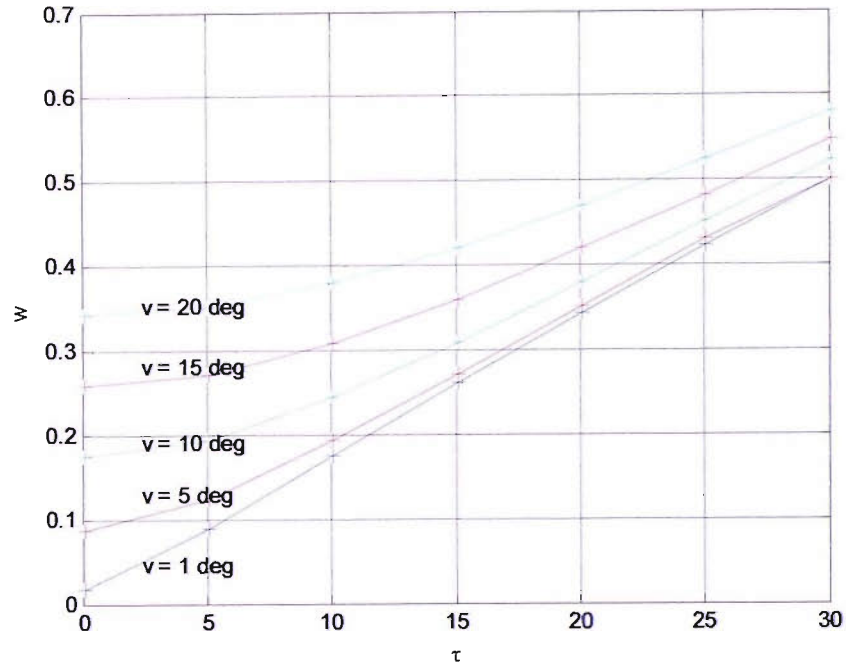


Figure 5.3: Critical measure of camera motion (case 1).

5.6 Integrating Self-Calibration with Distortion Correction

Distortion correction to an image is an important issue, especially when there is a need to register virtual objects correctly. It will become more important if the AR camera is being used in critical areas such as in surgical planning or other medical applications where incorrect registration due to distorted images might harm the patient under treatment because of the incorrect data it produces. Therefore, during the process of camera calibration, distortion correction needs to be taken into consideration.

Some offline calibration techniques such as the one described in Abdel-Aziz and Karara [1971] and most self-calibration algorithms available [Trivedi 1988; Mendonca and Cipolla 1999; Lourakis and Deriche 2000; Pollefey et al. 1999; Dornaika and Chung 2001] make the assumption that the images

that they use for calibration are distortion-free or have been corrected in a separate process before the calibration. However, in an AR system that employs a camera with varying parameters and especially if the AR camera has a wide-angle lens or zoom lens, the distortion can be varied between views, which make it easier for the distortion to be considered during calibration and not before calibration.

Therefore, in this section we develop an improved version of self-calibration in AR that has distortion parameters estimated simultaneously with the estimation of fundamental matrix whenever there is a need to correct distorted images. This enables the distortion parameters to be estimated online together with the camera parameters and these two processes can be done without the need of special calibration object. We follow the distortion model described in Chapter 2 combined with the algorithm proposed by Zhang [1996] to solve for distortion parameters and fundamental matrix and consequently intrinsic parameters.

5.6.1 Algorithm

Recall from Chapter 2 that lens distortion can be described by $\hat{u}_c = u_c + \delta_u$ (Equation 2.21) and $\hat{v}_c = v_c + \delta_v$ (Equation 2.22) where (u_c, v_c) are the distorted (true) image co-ordinates on the image plane and (δ_u, δ_v) are the distortion corrections to $(\hat{u}_c, \hat{v}_c)$. Recall that the transformation from point $\mathbf{U}_c$ to point $\mathbf{U}_a$ is given by $u_a = u_0 + \frac{u_c}{w}$ (Equation 2.4) and $v_a = v_0 + \frac{v_c}{h}$ (Equation 2.5) respectively where (u_0, v_0) is the principal point, w and h are the distances between adjacent pixels in the horizontal and vertical directions of the image plane respectively.

Combining Equation (2.21) with Equation (2.4) and Equation (2.24) with Equation (2.5) yields:

$$\hat{u}_c = w(u_a - u_0)[1 + (u_a - u_0)^2 k_1 w^2 + (v_a - v_0)^2 k_1 h^2] \tag{5.22}$$

$$\hat{v}_c = h(v_a - v_0)[1 + (u_a - u_0)^2 k_1 w^2 + (v_a - v_0)^2 k_1 h^2] \tag{5.23}$$

Since $(\hat{u}_c, \hat{v}_c)$ is the ideal co-ordinate expressed in the Euclidean co-ordinate system, we denote $(\hat{u}_a, \hat{v}_a)$ to be the ideal co-ordinate expressed in the image affine co-ordinate system, which is given by:

$$\hat{u}_a = \frac{\hat{u}_c}{w} \tag{5.24}$$

$$\hat{v}_a = \frac{\hat{v}_c}{h} \tag{5.25}$$

Equations (5.22) and (5.23) yield:

$$\hat{u}_a = (u_a - u_0)[1 + (u_a - u_0)^2 k_1 w^2 + (v_a - v_0)^2 k_1 h^2] \tag{5.26}$$

$$\hat{v}_a = (v_a - v_0)[1 + (u_a - u_0)^2 k_1 w^2 + (v_a - v_0)^2 k_1 h^2] \tag{5.27}$$

Equation (5.26) and (5.27) describe explicitly how ideal point co-ordinates are obtained from distorted ones.

Based on epipolar constraint, $\hat{\mathbf{U}}_a^T \mathbf{F} \hat{\mathbf{U}}_a' = 0$ (Equation (2.32) in Chapter 2), we can estimate k_1 provided that variables w and h are known where $\hat{\mathbf{U}}_a = [\hat{u}_a, \hat{v}_a, 1]^T$ and $\hat{\mathbf{U}}_a' = [\hat{u}_a', \hat{v}_a', 1]^T$.

5.6.2 Experiments on Radial Distortion

In our experiments we use a pair of distortion-free real images, *toto1.bmp* and *toto2.bmp*. We assume that variable w and h are known. We distort the images by applying different values of distortion parameter k_1 ranging from 1×10^{-6} to 2.5×10^{-5} with 1×10^{-6} for each step. An example of distorted corner distorted with $k_1 = 1\times10^{-5}$ is shown in Figure 5.4. The distorted images are then corner detected and correspondences between corners are found.

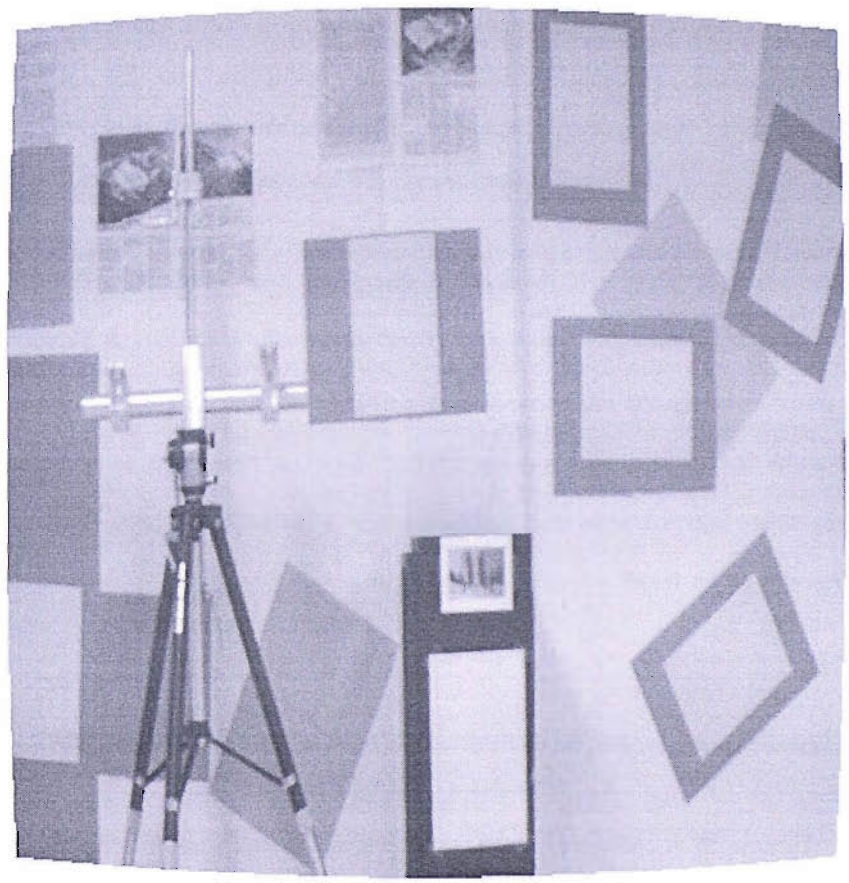


Figure 5.4: Example of *toto1.bmp* image distorted with value of $k_1 = 1\times10^{-5}$.

Two experiments are implemented. The first experiment includes estimation of variable k_1 only with the centre of distortion (u_0, v_0) set to be half of the size of the image vertically and horizontally. The second experiment consists of the estimation of variable k_1 together with the centre of distortion (u_0, v_0).

Figure 5.5 illustrates the comparison between the actual and estimated value of k_1. The algorithm can be said to estimate k_1 near to the actual value when the distortion is less than 7×10^{-5} and starts to be unstable when more severe distortion is imposed. The author believes that this is due to the number of correct point correspondence matches being reduced significantly as the distortion level is increased. This consequently affects the robustness of the fundamental matrix estimation stage.

Figure 5.6 illustrates the comparison between the estimated and actual values of k_1 with the distortion centre not fixed and allowed to vary. Our algorithm seems to be able to estimate k_1 near to the actual value when the distortion is less than 1.2×10^{-4}. This means that when the centre of distortion is allowed to vary, the algorithm can estimate the value of k_1 more reliably than when the centre of distortion is fixed for a distortion level less than 1.2×10^{-4}.

The value of the estimated distortion centre can be said to be stable and not vary too much from the actual principal point. The deviation of the average value from the true point is about 5.4%. This is illustrated in Figure 5.7.

It is worth mentioning here that, although the proposed algorithm can successfully estimate distortion parameters for medium to low distortion especially when the estimation of the distortion centre is included, the time

needed to estimate these parameters is enormous, around 30 minutes for estimation of k_1 and (u_0, v_0). The time to estimate k_1 parameter with a fixed (u_0, v_0) is around 3 minutes. Both experiments were implemented in Matlab. Therefore, it is suggested that the estimation of distortion parameters be done offline rather than online, especially when the distortion centre is also estimated. Additionally, if the distortion is not too severe, it is better to estimate the distortion parameter with a fixed distortion centre rather than allowing the centre to vary in order to take the advantage of the less completion time.

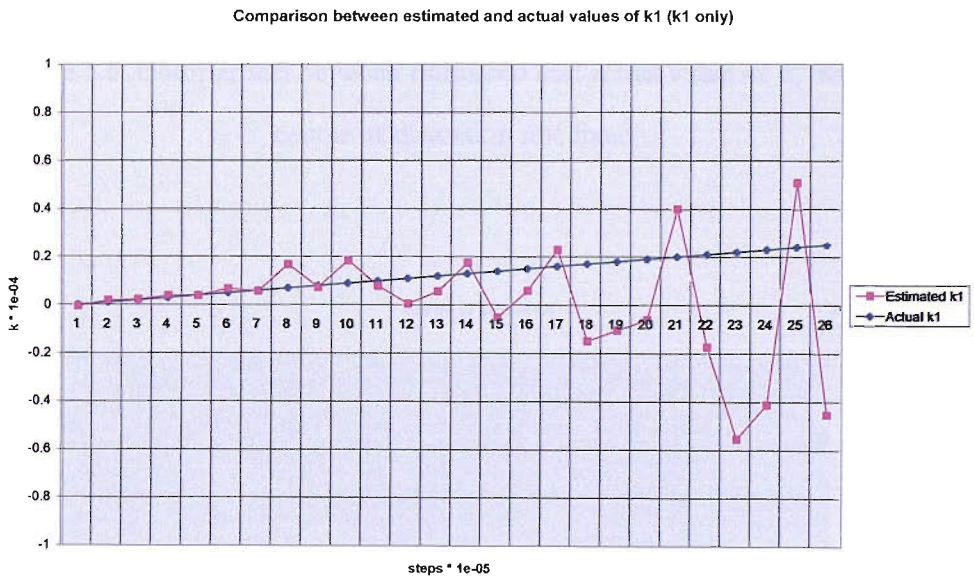


Figure 5.5: Comparison between estimated and actual value of k_1 with the centre of distortion fixed to half of the image.

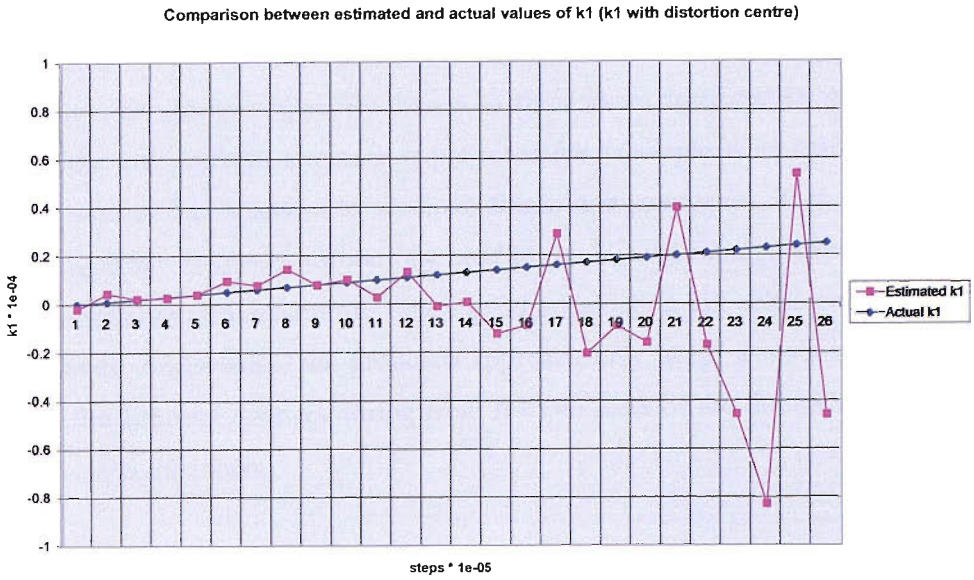


Figure 5.6: Comparison between estimated and actual value of k_1 with the centre of distortion not fixed.

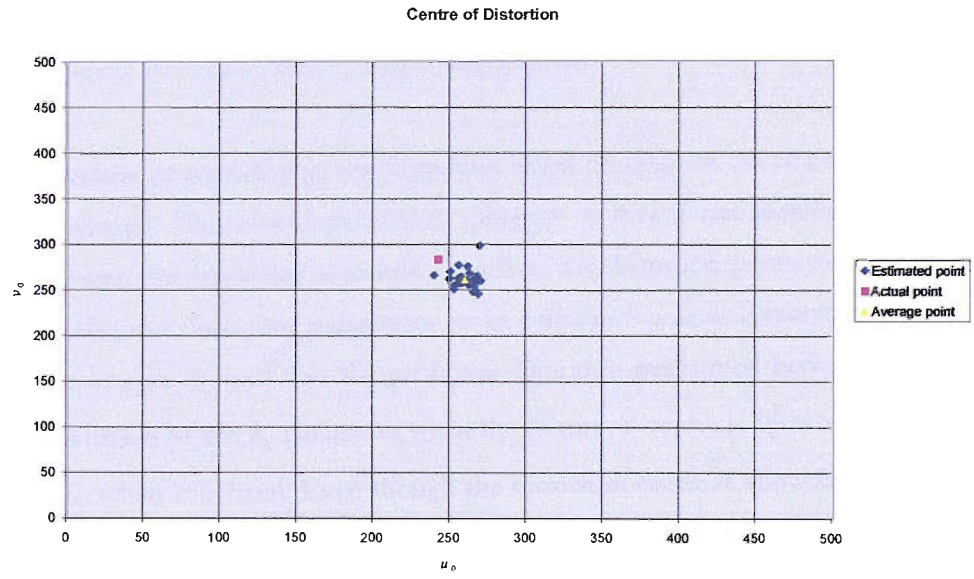


Figure 5.7: Distortion centre plot resulting from application of different distortion levels of k_1 1×10^{-6} to 2.5×10^{-5} with 1×10^{-6} for each step.

5.7 Conclusion

The fourth stage for self-calibration in AR has been described. The method employs an algebraic approach based on three views separated by general motion. The algebraic approach is proposed for its simplicity of derivation and requires less attention to accurate initialisation parameters. This shows the algorithm efficiency without the need to run eight-point algorithm prior to constraints minimisation to find a good initialisation like other available methods. Additionally, the proposed approach only needs a minimum of two fundamental matrices arising from the two pairs of the three views to be used as the inputs.

In this chapter, the common problem in self-calibration which is the critical motion is addressed. A new technique to measure the severity of camera motion involved in case 1 critical motion is also described. The measurement can be used to determine whether a pair of images can be chosen as inputs to be processed in the self-calibration stages. The technique proposed, however, only tackles one of the 3 critical motion configurations mentioned by Sturm [2002] and does not guarantee that the system is free from other critical configurations.

A technique to deal with lens distortion based on epipolar constraint is also developed. The advantage of this approach is that it can simultaneously estimate the fundamental matrix as well as the distortion parameters. This enables the distortion parameters to be estimated online. Experiments on real images showed that the proposed algorithm performed better on the estimation of the k_1 parameter when the distortion centre is allowed to vary than when it is fixed. Even though the distortion centre is allowed to vary, the estimated centre points are still close to the actual principal point, showing the robustness of the estimation of the distortion centre. The time required to estimate distortion parameters when the distortion centre

(u_0, v_0) is fixed is shorter than when the distortion centre is allowed to vary. Therefore, if the distortion is not too severe, it is advised to estimate only the k_1 term with a fixed distortion centre to take advantage of the shorter completion time. A disadvantage of the algorithm is that, it performs badly when the level of distortion is too severe $(k_1 > 1 \times 10^{-4})$. The condition of this high distortion however is rarely occurring in AR application.

Chapter 6

EXPERIMENTS

6.1 Introduction

This chapter describes experiments conducted based on the combination of the entire algorithm discussed in previous chapters. Let the combined algorithm be known as the Self-Calibration of Augmented Reality (SCAR) system. This chapter compares the performance of the SCAR system in terms of accuracy and stability with the ARToolKit, the latter being one of the most widely used AR systems. Experiments show that the SCAR system produces comparable results for intrinsic parameters to the offline calibration used in ARToolKit. The experiments also demonstrate its stability in producing consistent results for a sequence of frames at different time instances when the focal length is not changing. As the focal length changes, the system demonstrates its ability to estimate new intrinsic parameters, showing its adaptability to changes.

One of the differences between SCAR and ARToolKit is that the SCAR system proposes the calibration of intrinsic parameters online, which ARToolKit does not have; this is an advantage should the focal length of the camera change during an AR task. In contrast, ARToolKit suggests the use of offline camera calibration in an AR system, which has the benefit of

good accuracy of the camera's intrinsic parameters at the expense of being a time-consuming calibration process. In this chapter, the performance of both systems in terms of accuracy and stability will be presented and the suitability of each method to AR applications will be discussed. In the next section, we describe the procedures followed in our experiment for offline camera calibration in the ARToolKit and the steps taken for the SCAR system to show the flexibility of our system. Next, we measure the performance of both methods in terms of its accuracy of intrinsic parameter estimation. Then, we compare the stability performance of the SCAR system with real image data from a static camera and a video camera with and without distortion correction.

We then discuss other implementation issues related to the SCAR system in terms of the maximum iterations needed for the optimisation process and the sensitivity of the SCAR system to varying initialisation parameters for optimisation. The integration of the SCAR system into ARToolKit is also performed to show its applicability for easy incorporation into any AR system.

6.2 Offline Calibration in ARToolKit

In this section we describe the procedures of the offline calibration as suggested by the ARToolKit. It aims to show the accuracy and stability of the offline calibration used by the ARToolKit for an AR system. Two stages are involved, which include procedures for distortion parameter estimation and procedures for intrinsic parameter estimation. The suitability of these procedures to future AR systems is discussed. Results are presented and discussed at the end of the section.

6.2.1 Experimental Setup

The experiment was conducted using a Pulnix TM-765 camera with a COSMICAR television lens connected to a Hauppauge WinTV card with a Pentium III 700 MHz as the processor speed.

There are two calibration patterns employed in the ARToolKit camera calibration routines, *calib_dist* pattern and *calib_cparam* pattern, as shown in Figure 6.1 and Figure 6.2.

The *calib_dist* pattern contains an array of 6×4 dots, which is scaled so that the dots are exactly 40mm apart. This pattern is used to measure the amount of distortion produced by the lens. The *calib_cparam* pattern is a grid of lines and is scaled so that the lines are exactly 40mm apart. This pattern is used for the estimation of the intrinsic parameters after taking into account the distortion that may exist in the incoming images.

These two patterns were printed out from the *calib_cpara.pdf* and *calib_dist.pdf* files that come with ARToolKit and were glued to pieces of cardboard to make them flat and rigid.

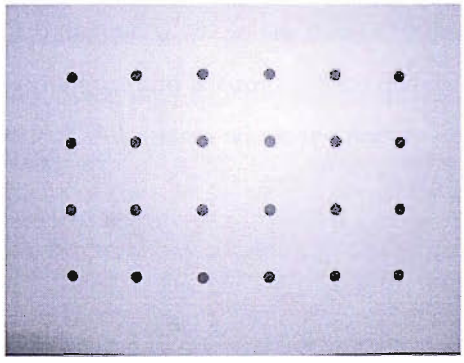


Figure 6.1: *calib_dist* pattern

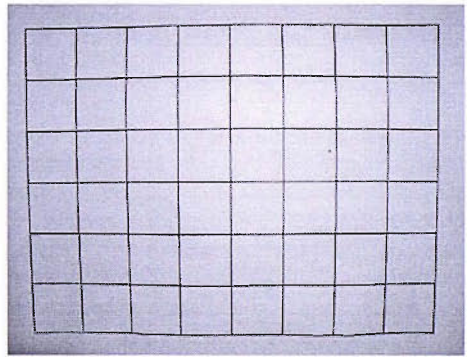


Figure 6.2: *calib_cparam* pattern

There are two programs that need to be run to calibrate the camera: firstly, the *calib_dist* program, which is used to measure the distortion centre of the camera image (x_0, y_0) and the distortion parameter k_1; and secondly, the *calib_cparam* program, which produces the calibration matrix of the camera. The *calib_dist* program was performed before the *calib_cparam* since the *calib_cparam* needs parameters from the *calib_dist* as its inputs.

6.2.2 Running the *calib_dist* Program

This program uses the co-ordinates of the dots from the *calib_dist* pattern as its input. The camera captures the image of the *calib_dist* pattern and the user will manually draw a rectangular shape around each dot in the following order as shown in Figure 6.3. The process will be repeated using other images taken from various positions and orientation of the same *calib_dist* pattern. The experimental setup is illustrated in Figure 6.4.

In this experiment, one of our aims is to verify that the greater the number of images taken, the more accurate the calibration, as is mentioned in the ARToolKit documentation.

All calibration steps are manually done by the user, which can take quite some time and is cumbersome due to the repetition. This is in contrast with our SCAR system where the feature detection can be done automatically.

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Figure 6.3: The sequence of dots to be covered by the user for *calib_dist* pattern.

6.2.3 Results and Discussion for *calib_dist*

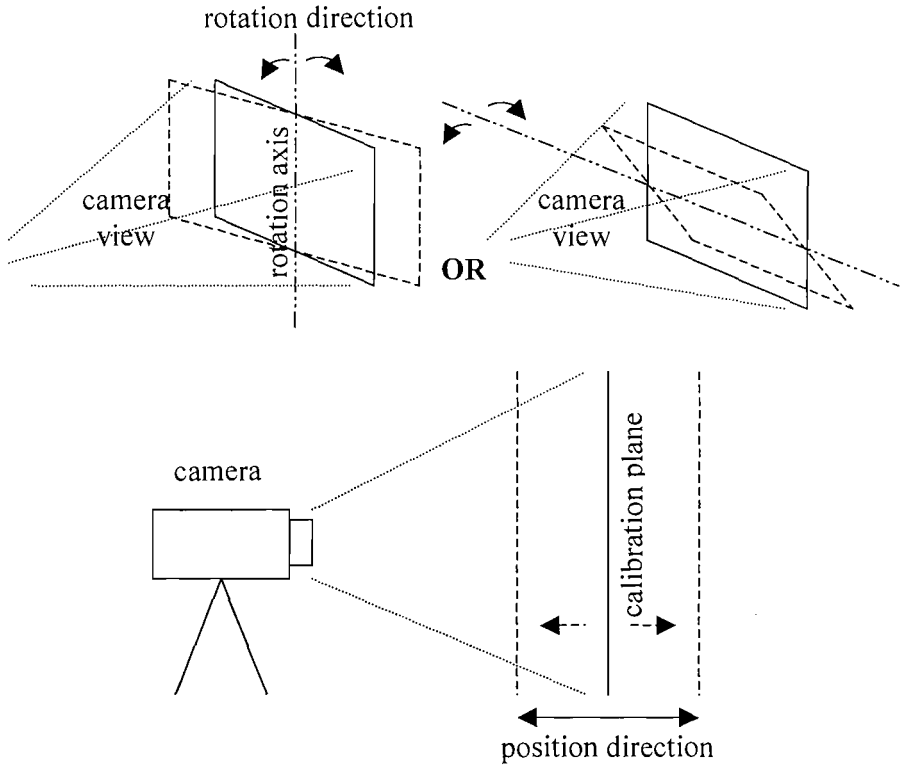


Figure 6.4: Experimental setup for *calib_dist* program.

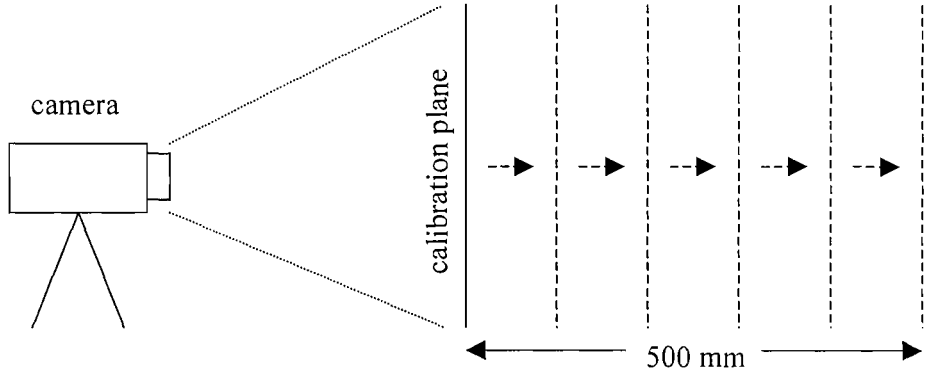


Figure 6.5: Experimental setup for *calib_cparam* program.

When running the *calib_dist* program, after completing the manual steps in the previous section, the program will automatically store the co-ordinates of each dot in memory. Table 6.1 shows an example of the co-ordinates stored in memory for a captured image. In this case, the position of the pattern's plane is perpendicular and at a distance of 45 cm from the camera. As we can see in Figure 6.6, the rows and columns of the dots are not in straight lines, which is due to camera lens distortion.

After several images of different positions and orientations have been captured, the program calculates the distortion centre of the image (u_0, v_0) and the distortion parameter k_1 in order to correct any distortion exhibited from the lens. Table 6.2 shows the end results from *calib_dist*.

From Table 6.2, we see that the distortion centres do not converge to a certain value as the number of captured images increases. In other words, the number of captured images has little to do with producing a better result. Once these values have been passed to the next program (*calib_cparam*), this will result in different values for the calibration matrix parameters.

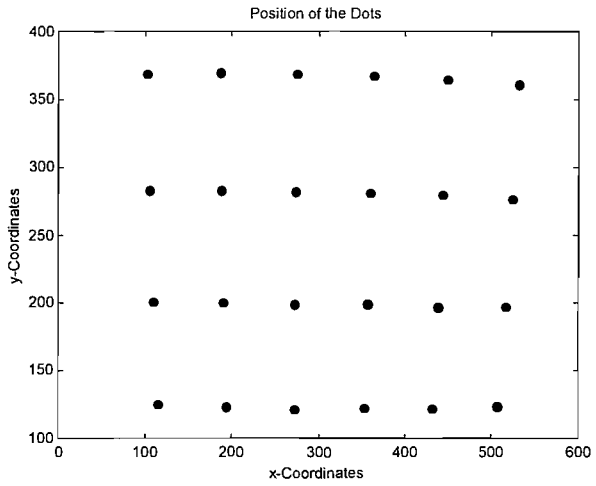


Figure 6.6: Plotted co-ordinates of the dots.

Table 6.1: The Co-ordinates of All 24 Dots

Row	Column	Co-ordinate X (pixel)	Co-ordinate Y (pixel)
1	1	113.85	125.25
2	1	191.97	123.82
3	1	271.76	122.23
4	1	351.98	122.61
5	1	430.61	122.81
6	1	506.06	123.86
1	2	107.98	201.04
2	2	189.46	200.51
3	2	272.26	198.98
4	2	355.83	199.02
5	2	437.39	197.88
6	2	515.58	197.62
1	3	103.60	282.78
2	3	187.56	283.24
3	3	273.00	282.14
4	3	359.30	281.24
5	3	443.43	279.33
6	3	524.11	277.15
1	4	100.57	368.89
2	4	186.15	369.88
3	4	274.03	369.59
4	4	362.43	367.84
5	4	448.63	364.89
6	4	531.14	360.90

Table 6.2: Results from *calib_dist*

Number of Images Captured	Distortion Centre, u_0 (pixel)	Distortion Centre, v_0 (pixel)	Distortion parameter, k_1 (multiplied by 10^8)
10	301.5	211.0	52.5
9	298.5	222.0	51.0
8	294.5	219.0	53.0
7	299.0	221.5	51.8
6	278.0	231.5	45.8
5	305.5	225.5	51.6
4	299.0	206.5	53.1
3	306.5	225.5	52.3
2	303.5	229.0	54.0
1	304.0	217.0	53.7

6.2.4 Running the *calib_cparam* Program

This program uses a grid pattern of 7 horizontal lines and 9 vertical lines from the *calib_cparam* pattern as its input (refer Figure 6.2). The procedures for *calib_cparam* program are as follows:

a) Type *calib_cparam* at the command prompt and input the distortion centre co-ordinates and distortion parameter k_1 when requested. A live video appears.
b) Place the grid pattern in the camera view so that the pattern is perpendicular to the camera with all of the grid lines in view.
c) Click the left mouse button to capture the image. This generates a white horizontal line overlaid on the image.

d) Move the computer-generated line to fit the top-most horizontal grid line. The arrows keys are used to move the line up, down, clockwise or anticlockwise. As the white line is aligned with the top-most horizontal grid line, press the enter key. This line turns blue and another white line is generated. This process should be repeated for all the horizontal lines from top to bottom.

e) When the last horizontal line has been placed, a vertical white line appears and process (d) is repeated for the vertical grid lines from left to right.

f) Once this process is completed for one image, the grid pattern is moved 100mm away from the camera (keeping the camera perpendicular to the pattern). Then, the whole process (a) to (e) is repeated 5 times until the total distance of plane movement away from the camera becomes 500mm.

g) After the fifth movement, the program automatically calculates the camera parameters. The user is prompted for a filename for storing these parameters in memory.

Figure 6.5 illustrates the experimental setup for the above-mentioned processes.

6.2.5 Results and Discussion for *calib_cparam*

Based on the values in Table 6.2, which are passed into the *calib_cparam* program, the following results shown in Table 6.3 were obtained. These results were drawn in a chart as shown in Figure 6.7 and Figure 6.8.

Table 6.3 illustrates the intrinsic camera parameters produced by the *calib_cparam* program. In Figure 6.7, it is shown that the values for the centre co-ordinates vary between the range of 200 and 300 as the number of captured images in *calib_dist* increases. The same case applies for the value of

α_u and α_v where the values keep changing between the range 800 to 900 (refer to Figure 6.8). These figures show that it is difficult to get accurate results even though the number of captured images has been increased. This can happen due to the number of images being captured involving user intervention, which increases with the number of images. Therefore more human error is likely to occur during the calibration process.

Suppose we take the average value to be the true value, the standard deviation of the image centre is equal to 6.8% from the true value. As for the scale factors, the standard deviations for α_u and α_v are both 3.5%. These values will be compared with the one produced by the SCAR system in Section 6.3.

Table 6.3: Results from *calib_cparam*

Number of Captured Image	α_u	α_v	u_0	v_0
10	796.241	818.242	279.637	232.433
9	861.804	882.457	258.062	233.239
8	847.152	866.309	277.637	272.392
7	832.968	855.019	246.272	262.016
6	807.511	832.661	216.184	262.064
5	834.73	859.495	241.259	254.637
4	821.739	844.389	260.869	223.386
3	881.247	906.183	230.814	253.543
2	843.687	869.928	204.533	193.674
1	787.377	807.932	281.265	267.265
Average	**831.446**	**854.262**	**249.653**	**245.465**

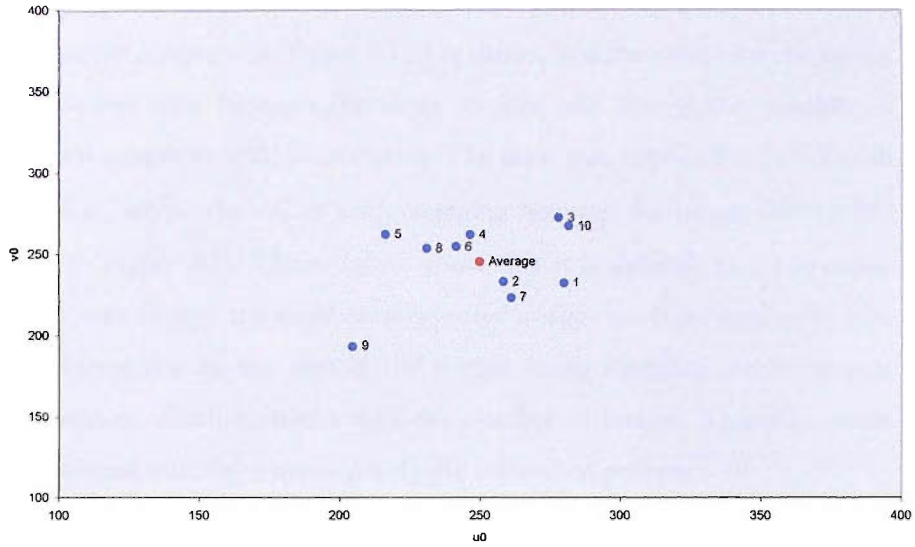


Figure 6.7: Image centre point resulting from *calib_cparam* for different number of captured images.

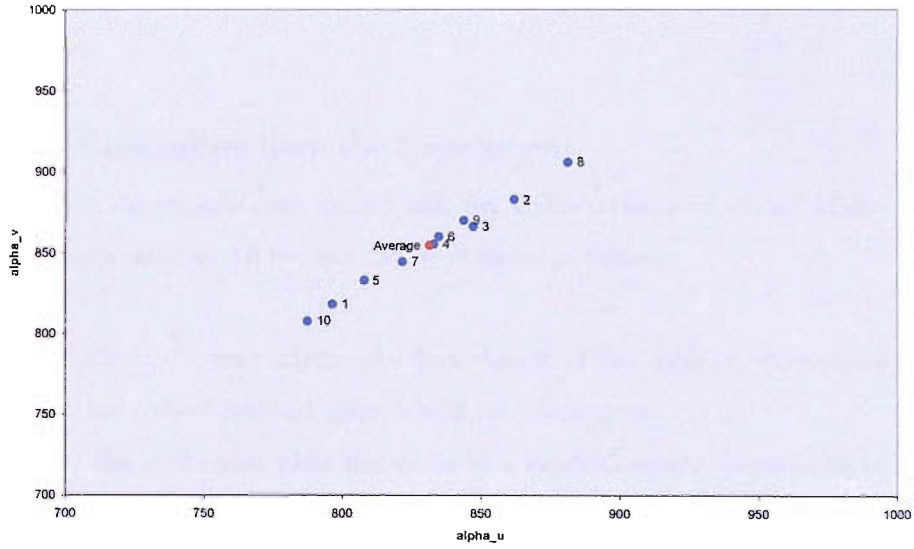


Figure 6.8: α_u and α_v parameter from *calib_cparam* for different number of captured images.

Table 6.3 illustrates the intrinsic camera parameters produced by the *calib_cparam* program. In Figure 6.7, it is shown that the values for the centre co-ordinates vary between the range of 200 and 300 as the number of captured images in *calib_dist* increases. The same case applies for the value of α_u and α_v where the values keep changing between the ranges 800 to 900 (refer to Figure 6.8). These figures show that it is difficult to get accurate results even though the number of captured images has been increased. This can happen due to the number of images being captured involving user intervention, which increases with the number of images. Therefore more human error is likely to occur during the calibration process.

Suppose we take the average value to be the true value, the standard deviation of the image centre is equal to 6.8% from the true value. As for the scale factors, the standard deviations for α_u and α_v are both 3.5%. These values will be compared with the one produced by the SCAR system in Section 6.3.

6.2.6 Conclusion from the Experiment

Based on the experiments carried out, the characteristics of offline plane-based calibration in ARToolKit can be assessed as follows:

a) Once the user adjusts the focal length of the camera, the camera has to be calibrated again, which can waste time.
b) The calibration plate has to be in a location where the amount of light is sufficient and the reflection of the light on the camera is kept to a minimum.
c) The calibration process involves human manipulation, which is prone to error.
d) Since the *calib_dist* pattern is positioned randomly before capturing the image, the result is in some way dependent on the position and

orientation of the pattern plane. Results for intrinsic parameters appear to be different with different setups for the plane orientation and position. For example, results for patterns captured slanting to the right are different from those of patterns captured slanting to the left.

6.3 Experiment on Self-Calibration Augmented Reality (SCAR) System

In this section, the SCAR system was evaluated in terms of stability and the accuracy of the intrinsic parameters produced. Using the same camera, the accuracy of the intrinsic parameters produced by the SCAR system is compared with the intrinsic parameters resulting from the offline camera calibration in ARToolKit. In addition, using the same camera, the stability of the values of intrinsic parameters from different captured views is also compared with that of the values from the offline calibration in ARToolKit.

Another criterion that needs to be compared in the SCAR system is perhaps its performance in terms of speed. Since the prototype of the SCAR system is done in Matlab, the actual performance of the algorithm in terms of speed cannot be compared with that in ARToolKit. It goes without saying that by doing such calibration offline the calibration in ARToolKit will always outperform any self-calibration embedded in an AR system, especially when there is no need to adjust the focal length during an AR task.

As speed is one of the important criteria for real-time AR systems, we propose not to process every frame in order to update the camera's intrinsic parameters. Instead, in our implementation, we select one frame for every second. This frame will then be corner-detected and the total number of correct point correspondence matches checked. If the total of correct

correspondences is more than 20 matches, then the frame is selected for further processing, otherwise we discard it. Then, if the motion involved is a critical one, the next frame will be selected. This procedure is repeated until three views are collected. Following this, the first intrinsic parameters are estimated – this is currently done for video scene.

After this the system will search for another three views from the next sequence of frames so that any changes in focal length **between** the former and the current three views can be updated as soon as possible. However, our system assumes there are no changes in focal length **within** the three captured views to be processed. This is for simplicity, which is due to the algorithm derivation (refer to Chapter 5, Section 5.4), which has $4 + 4n$ number of unknowns and $7n$ number of constraints, where n represents the number of motions involved in solving them. If we were to consider the changes of focal length **within** the three captured frames in the algorithm, the number of unknowns and constraints to be solved would be $4 + 4n + \mathbf{4}s$ and $7n$, respectively. Hence this would require a minimum of 4 views or 3 motions to recover the unknowns within the 4 views. The additional $\mathbf{4}s$ unknowns represent the changes in the **4** intrinsic parameters of the s additional views.

6.3.1 Experiments for Intrinsic Parameters from Static Camera

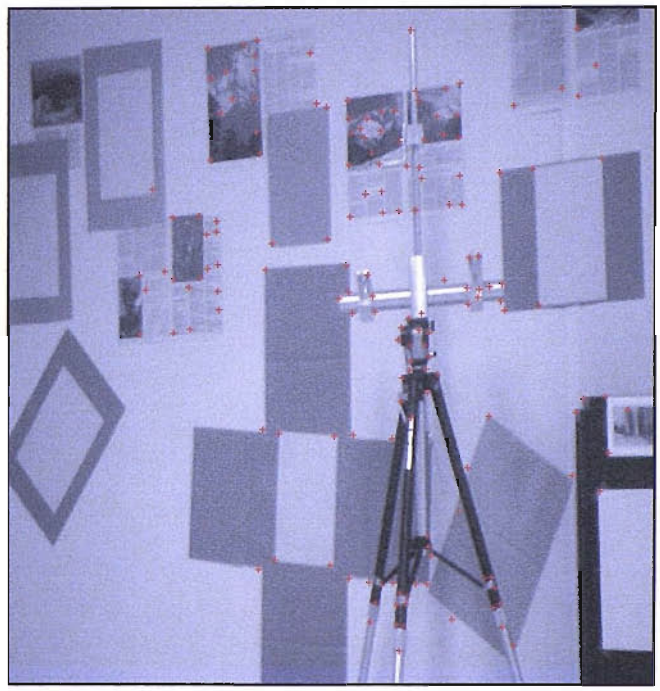
In this experiment, no camera is used; instead, test images with three views acquired from several websites were used. This is due to the actual values provided for the intrinsic parameters, which can be used as a reference to the outcome of the experiments carried out for our method. These values were used as a basis to check the performance of the SCAR system in terms of its accuracy and stability.

The experiments were carried out on sets of monocular test images. The test images were retrieved from the public web site ftp://ftp-robotvis.inria.fr/pub/IMAGES ROBOTVIS. Figure 6.9(a) and Figure 6.9(b) illustrate examples of the two pairs of test images taken from a static camera. The matches are shown as red crosses. The results from self-calibration methods are shown in Table 6.4.

The first row of Table 6.4 gives the initialisation of the non-linear methods in self-calibration. Note that these values are chosen based on good guesses. The second row shows the correct intrinsic parameters associated with the camera capturing the images as provided by the website. The third row shows the intrinsic parameters acquired from the algebraic method in our SCAR system. The values shown are the average from 10 repetitions. Note that from Table 6.4 the self-calibration produces results that are close to the reference solution. The deviation of α_u and α_v from the true value is 2.8% and 1.4% respectively whereas for the centre point it is 7%.

Table 6.4: Self-Calibration Results

Method	α_u	α_v	u_0	v_0	Iterations
Initialisation	1500	1500	250	250	-
Reference (Off-line calibration)	659	935	242	283	-
Self-Calibration	640.64	948.30	258.68	286.38	32
Deviation	2.8%	1.4%	7.0%		



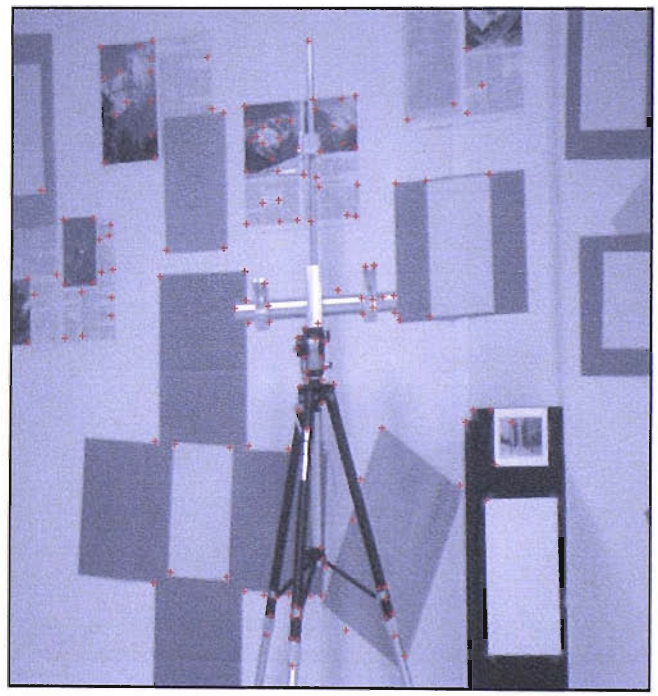


Figure 6.9(a): First pair of test images from static camera.

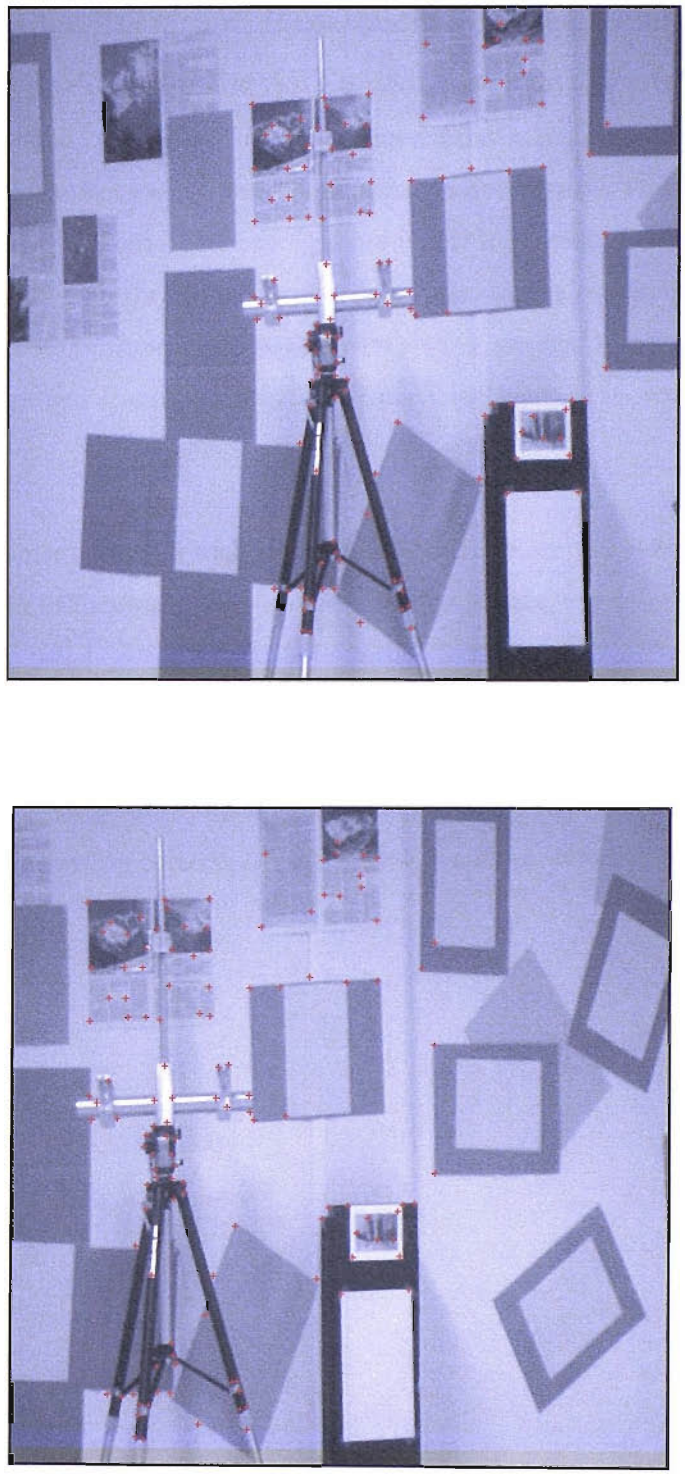


Figure 6.9(b): Second pair of test images from static camera.

6.3.2 Radial Distortion Experiments on Test Images from Live Video for the ARToolKit and SCAR System

In this section, test images were generated from a sequence of captured frames from a live video camera connected to a computer. The difference between images from a static camera and live video is that the former have better quality than the latter. In other words the images captured from live video have more noise than images from a static camera. Examples of test images captured from the COSMICAR television lens are shown in Figure 6.14.

An experiment with 20 trials was carried out to check the consistency of camera self-calibration results on the captured images and to compare the effect of radial distortion on the result from camera self-calibration. Comparison between the results from this experiment is based upon the results from the offline camera calibration. Distorted images were corrected offline and the intrinsic parameters resulting from self-calibration before and after distortion correction were recorded and are shown in Figure 6.10 to Figure 6.13.

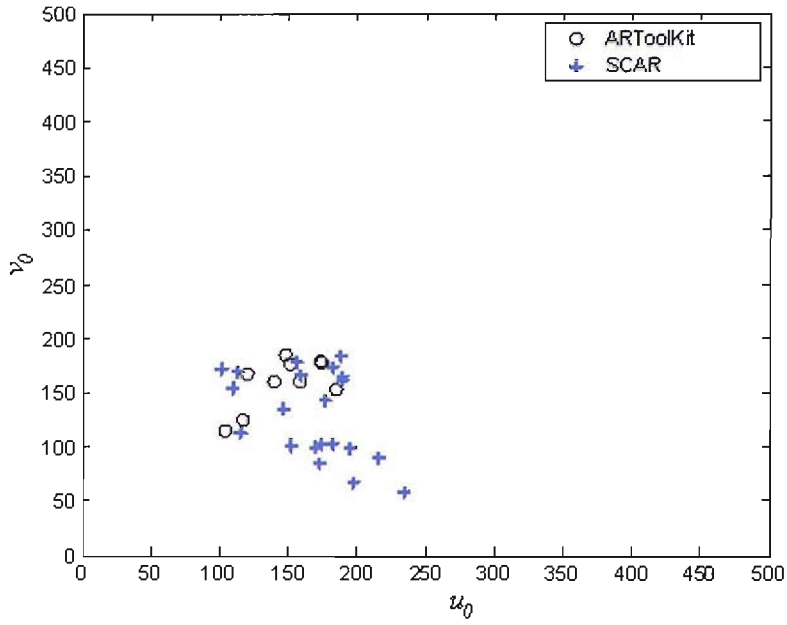


Figure 6.10: Principal points without distortion correction.

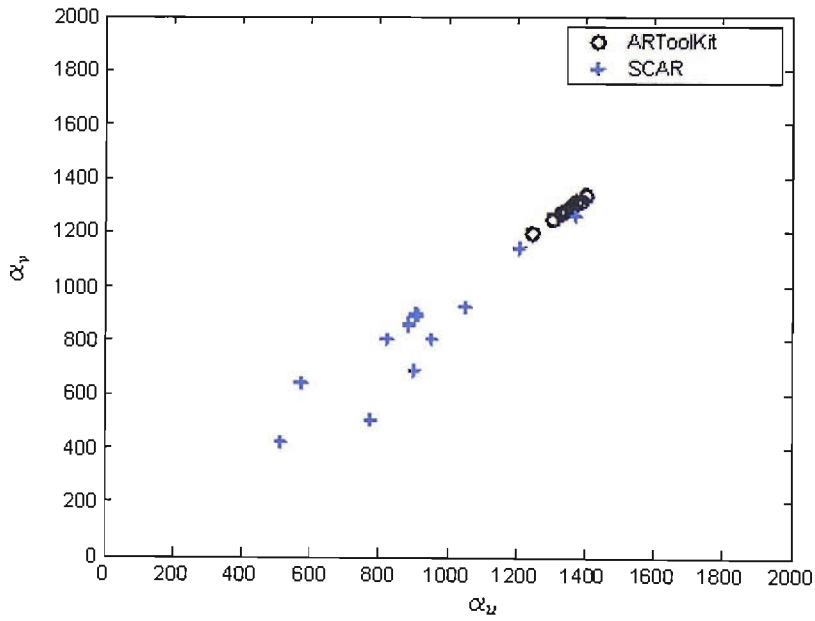


Figure 6.11: Scale factor without distortion correction.

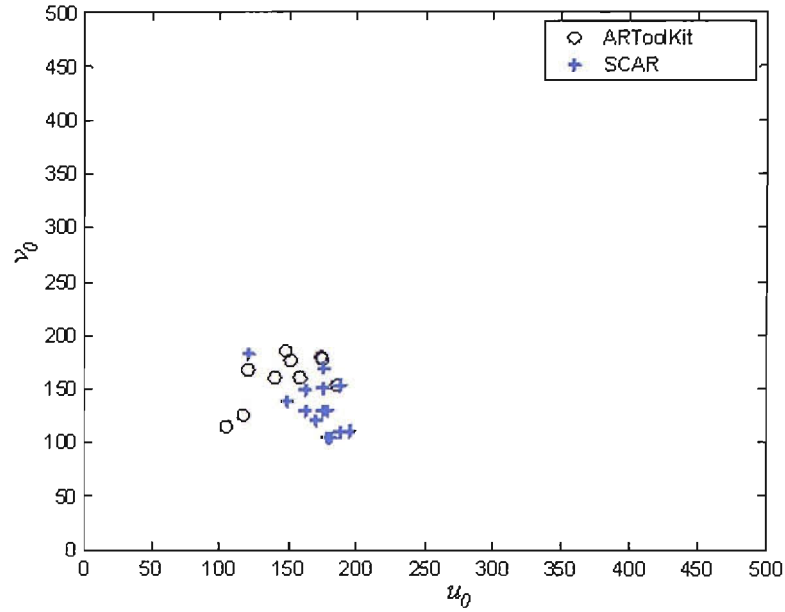


Figure 6.12: Principal points with distortion correction.

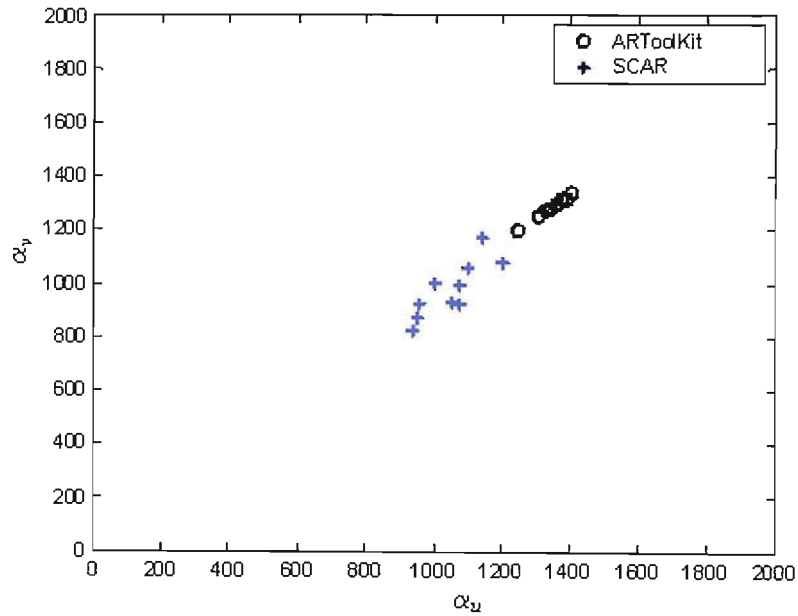
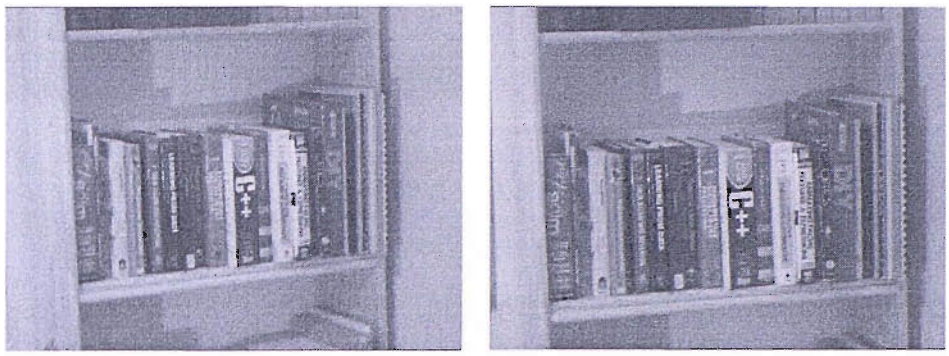


Figure 6.13: Scale factor with distortion correction.



(a) (b)

Figure 6.14: Example of test images captured from live video.

From Figure 6.10 and Figure 6.11, we can observe that the principal points and scale factors estimation for self-calibration without distortion correction is not very reliable for noisy images. However, Figure 6.12 and Figure 6.13 show improvements in the clusters for the self-calibration method in comparison to the clusters in Figure 6.10 and Figure 6.11. From this we can presume that taking distortion correction into account will improve the accuracy of the camera calibration parameters for noisy images.

6.4 Full Implementation of the SCAR System

The goal of this experiment is to examine implementation issues that may arise when the SCAR system is implemented as a whole including the pre-calibration stages. In order to achieve this, 3 different sequences (*toto, highouse_frame and fountain*) in Appendix B have been chosen as the test sequences. The dimensions of *toto, highouse_frame* and *fountain* sequence are 512×512, 576×384 and 320×240 pixels respectively.

The experiment takes these three uncalibrated sequences as inputs and implements them using the following procedures:

A. Corner Detection using Harris-SUSAN Hybrid Corner Detector (refer to Chapter 3).

a. The algorithm will detect corners from each image. The following parameters are set:
    - Standard deviation of smoothing Gaussian, $\sigma = 3$.
    - Threshold value for Harris detector, $thresh = 1000$.
    - Radius of region considered in non-maximal suppression (optional), $radius = 3$.

B. Point Correspondence Matching (refer to Chapter 4).
    a. Matching is done using correlation and calculating motion vector. Point is said to be matched when the correlation between them has the maximum value.
    b. False matches and outliers are discarded by using *reference vector.*

C. Fundamental Matrix Estimation
    a. A modification of the MAPSAC algorithm based on controlled random selection of corner points is used.

D. Intrinsic Parameters Estimation based on three views (refer Chapter 5)
    a. An algebraic method with Levenberg-Marquardt technique for minimisation process is used.

Since all sequences are not distorted, an algorithm without distortion correction is used. The results from the intrinsic parameters estimation are plotted and are used as input to be integrated into ARToolKit. The findings are discussed in the following section.

6.4.1 Results from Correspondence Matching and Calibration Matrix

The following figures illustrate the results from corner detection as well as point correspondence matching. The total numbers of corners detected by the Harris-SUSAN corner detector before and after matching are recorded in Table 6.5, Table 6.6 and Table 6.7 for *toto*, *bighouse* and *fountain* sequence respectively.

Based on observation from Figure 6.15 to Figure 6.18, there is not a single false match after point correspondence matching. Thus, these correspondences can become good inputs for the fundamental matrix estimation stage.

From this experiment, it can be concluded that, provided the corners are well detected, the algorithm for matching can produce reliable correspondences while retaining sufficient number of good corners for fundamental matrix estimation. Accurate values for corners and correct matches are very important for fundamental matrix estimation due to the fact that it is very sensitive to any changes in corner positions and incorrect matches. This experiment demonstrates the robustness of our developed matching algorithm even though there are missed corners resulting from the Harris corner detector. The resulting calibration matrix for *fountain* sequence is $\alpha_u = 398.344$, $\alpha_v = 254.962$, $u_0 = 154.226$, $v_0 = 134.490$ and for *bighouse_frame* sequence is $\alpha_u = 976.676$, $\alpha_v = 1027.809$, $u_0 = 371.151$, $v_0 = 210.300$.

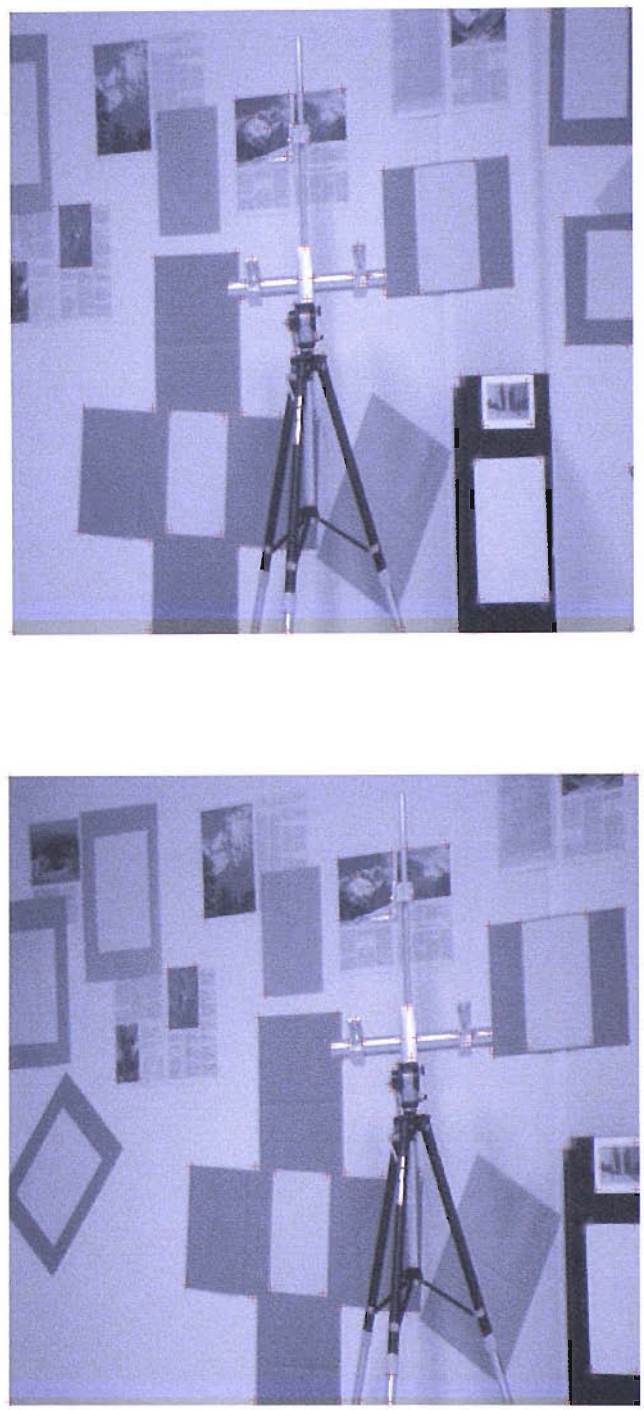


Figure 6.15: Corners detected for *toto2.bmp* and *toto3.bmp* before matching.

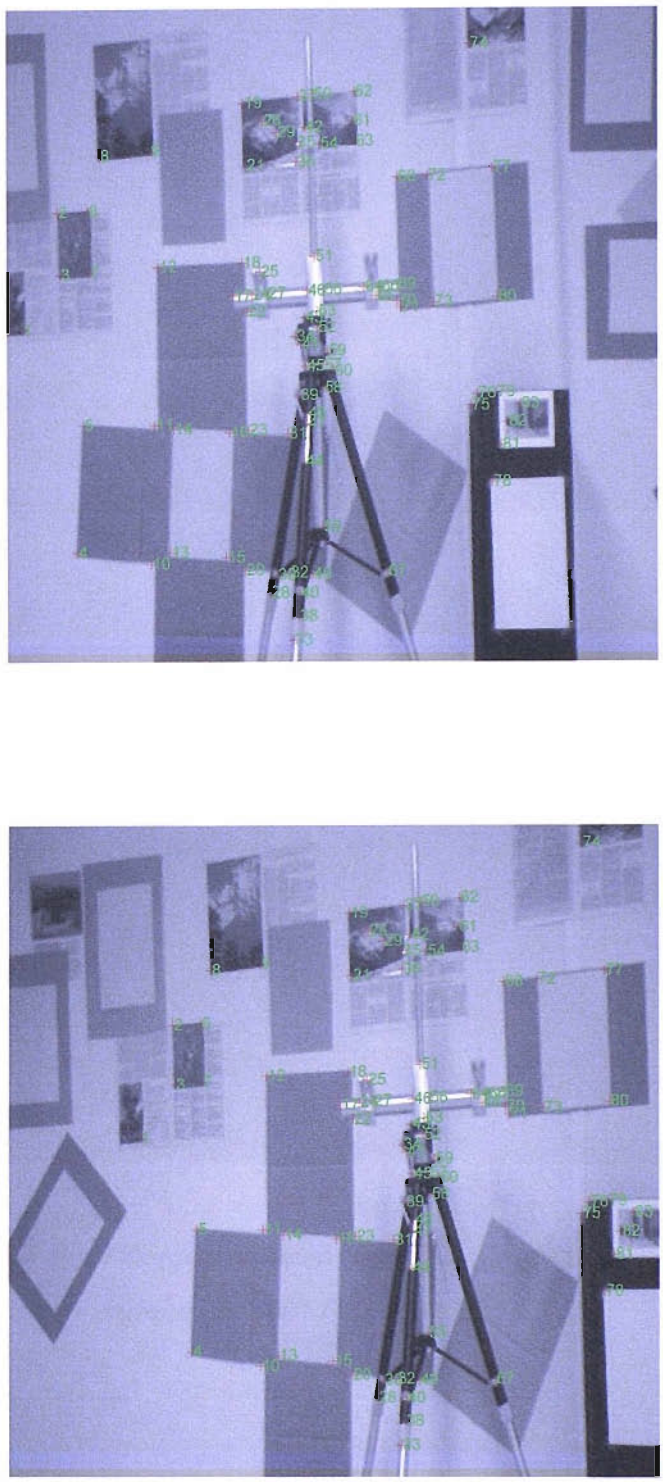


Figure 6.16: Corners detected for *toto1.bmp* and *toto2.bmp* after matching.

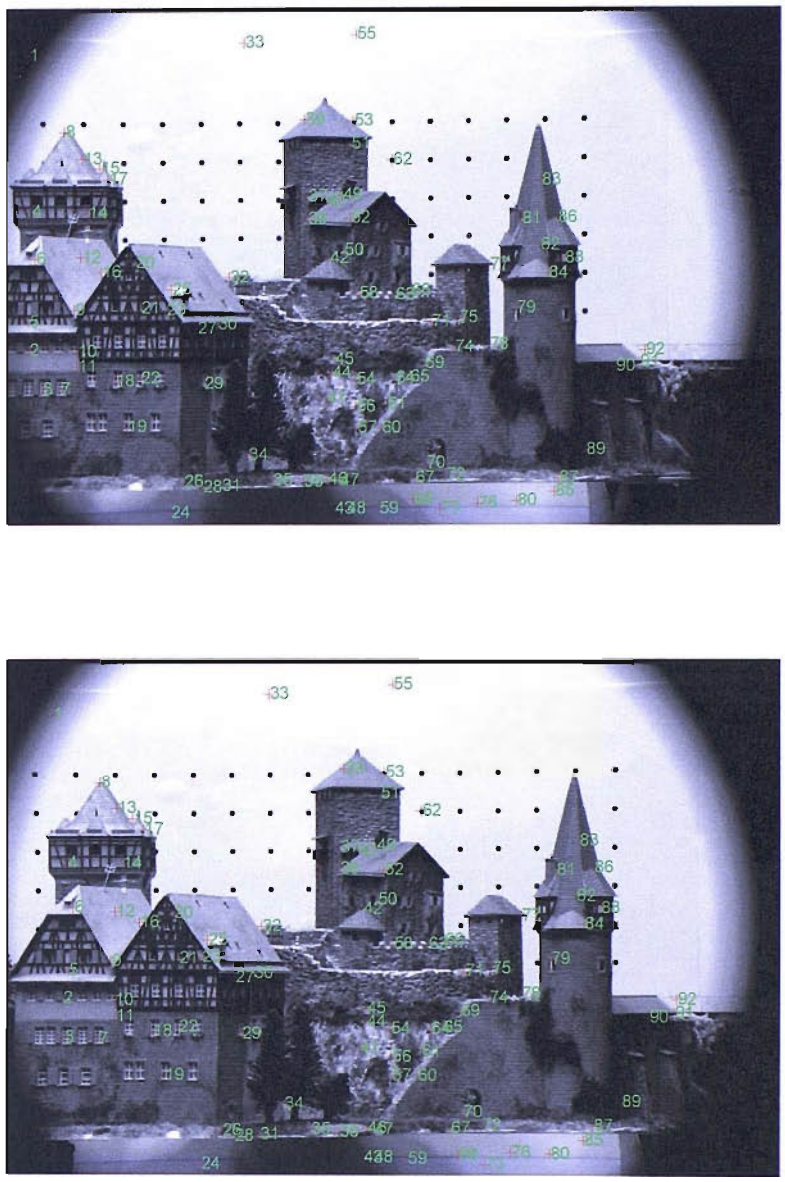


Figure 6.17: Corners detected for *bighouse_frame000.bmp* and *bighouse_frame001.bmp* after matching.

Figure 6.18: Corners detected for *fountain03.bmp* and *fountain05.bmp* after matching.

Table 6.5: Number of corners before and after matching for *toto* sequence.

	toto1	*toto2*	*toto3*
Total corners before matching	113	123	114
Total corners after matching	86		
		74	
Percentage of discarded corners	24%	30%-40%	35%

Table 6.6: Number of corners before and after matching for *bighouse_frame* sequence.

	bighouse_frame001	*bighouse_frame002*	*bighouse_frame003*
Total corners before matching	579	611	596
Total corners after matching	92		
		92	
Percentage of discarded corners	84%	85%	85%

Table 6.7: Number of corners before and after matching for *fountain* sequence.

	fountain01	*fountain03*	*fountain05*
Total corners before matching	187	168	183
Total corners after matching	120		
		113	
Percentage of discarded corners	36%	29%-33%	38%

6.4.2 Value Setting for Maximum Iterations in Levenberg-Marquardt Optimisation

Maximum Iterations is defined as the maximum number of iterations allowed for Levenberg-Marquardt optimisation to perform in order to find the actual value that minimise the constraints. The purpose of this experiment is to find the best value for Maximum Iterations that can produce the optimum result for the intrinsic parameters. In this experiment, the system runs up to 100 times with the same test image. The intrinsic parameters are recorded each time.

Figures 6.19 to 6.21 illustrate how the values for Maximum Iterations may affect the accuracy and consistency of the scale factors. The x-axis represents alpha_u (α_u), which is the distance between adjacent pixels vertically, and the y-axis represents alpha_v (α_v), which is the distance between adjacent pixels horizontally. Figures 6.19, 6.20, 6.21 and 6.22 represent the scale factors plot for Maximum Iterations of 200, 100, 50 and 30, respectively. From these results the optimum number of Maximum Iterations for the whole system to be accurate is 30. Any value set more or less than 30 for the Maximum Iterations will result in the estimation of the intrinsic parameters become less accurate. From the figures, we can see that as the Maximum Iterations value is decreased the cluster of points becomes more focused and the distance between the average and actual scale factors becomes closer.

Figures 6.23 to 6.26 illustrate how the value for Maximum Iterations affects accuracy and consistency for the principal points. The x and y axis represent the coordinates of the principal points (u_0, v_0) found in the image. Figure 6.23, 6.24, 6.25 and 6.26 represent the scale factor plot for Maximum Iterations of 200, 100, 50 and 30 respectively. From this result the optimum number of Maximum Iterations for the system to be accurate is 30 as is the case with the scale factors. From the obtained results, we can see that 30 is

the optimal value for Maximum Iterations in the Levernberg-Marquardt minimisation process in order to obtain the most accurate results for intrinsic parameters.

One might argue that the reason for the points grouping into clusters when Maximum Iterations is set to a reduced limit is that the author has chosen the initial value to be near the actual value. This is not the case, as is shown in Table 6.8 and Table 6.9. Even though the initialisation number is larger or smaller than the true value, the number of iterations taken to complete the minimisation is still about the same, that is, less than 30. Exceptions are when the initial values for α_u and α_v were chosen as less than 400, or $u_0 = v_0$ were greater than 1000, which rarely happens. This proves that the number of iterations taken to complete the minimisation is not highly affected by the initial value chosen.

Table 6.8 and Figure 6.27 demonstrate the stability performance of SCAR for different initialisation values of the scale factors α_u and α_v. In this case the initialisation value for the principal point is set to $u_0 = v_0 = 250$. The results show that values for the resulting intrinsic parameters are not affected by the different initialisation values set for α_u and α_v.

Table 6.9 and Figure 6.28 illustrate the stability performance of SCAR for different initialisation values of the principal points u_0 and v_0. In this case, the initialisation values for the scale factors are set to $\alpha_u = \alpha_v = 1500$. The results also show that the resulting intrinsic parameters are not affected by different initialisation values being set for u_0 and v_0. This proves the robustness of this algorithm compared to another available algorithm [Luong and Faugeras 1997], which requires near to true initialisation values in order to get correct results.

6.4.3 Computational Complexity and Practical Realisation

Table 6.10 shows the performance of the whole algorithm in terms of speed by using image with resolution 512 x 512 pixels. The algorithm was run in Matlab installed in a computer with a 320 Mb RAM memory and Pentium III 800 Mhz processor. It shows that the whole algorithm developed requires around 20 seconds to be completed. The fundamental matrix estimation consumes the most time followed by corner detection, calibration matrix estimation and point correspondence match. The longer time required by the fundamental matrix estimation is due to a high number of samples for correspondence matches to be processed. The lower this number is set by the user, the faster the time of completion. In this case, we set the number of samples to be 1000.

The CPU time required for calibration matrix estimation stage is 5.1 second for 30 iterations using Levenberg-Marquardt technique. For each iteration, the time required is generally higher than other self-calibration methods due to more constraints and unknowns to be solved in the algebraic approach. However, the need for additional time to obtain the correct initialisation parameters and the uncertainty of the number of iterations involved by other self-calibration methods make the algebraic approach to be our preferred method.

Table 6.11 illustrates the computational complexity for each stage involved in the SCAR system. The computational complexity of MAPSAC is not fixed because it depends on the number of sample matches set by the user. For practical realisation of SCAR, the whole code needs to be converted into C++ to take the advantage in terms of speed. Then, capturing of the 3 images required for SCAR should be set to be done every minute. Whenever there is a need for updating the intrinsic parameters in between the minutes, the AR system should be able to prepare an icon button where the user can simply update the parameters at any time.

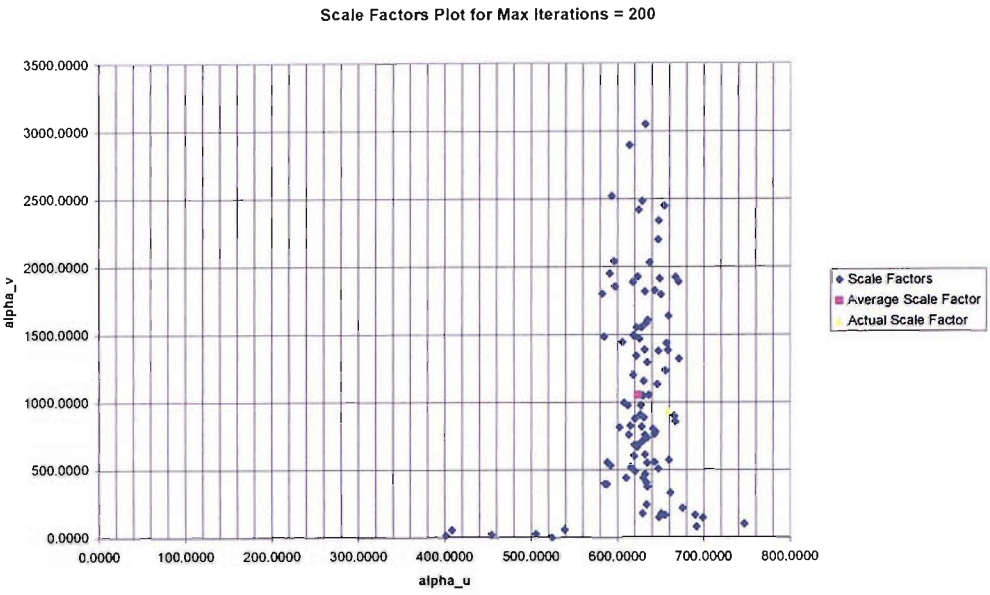


Figure 6.19: Scale Factors Plot where Maximum Iterations = 200.

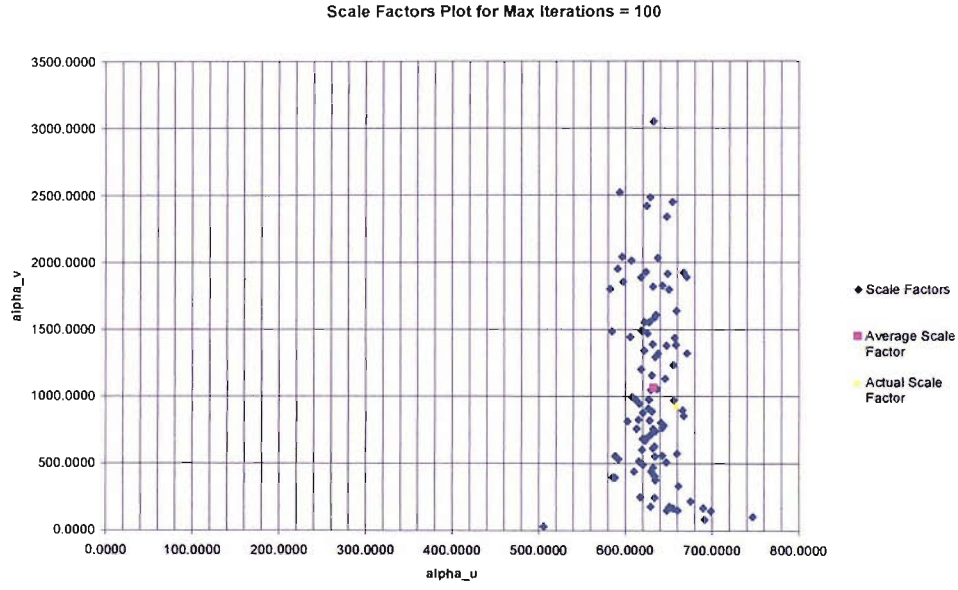


Figure 6.20: Scale Factors Plot where Maximum Iterations = 100.

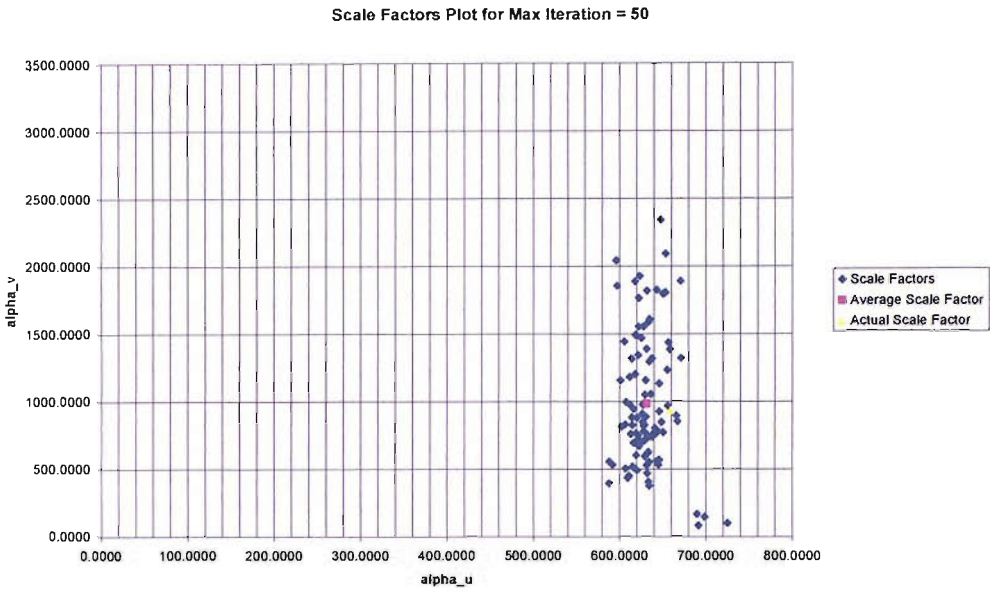


Figure 6.21: Scale Factors Plot where Maximum Iterations = 50.

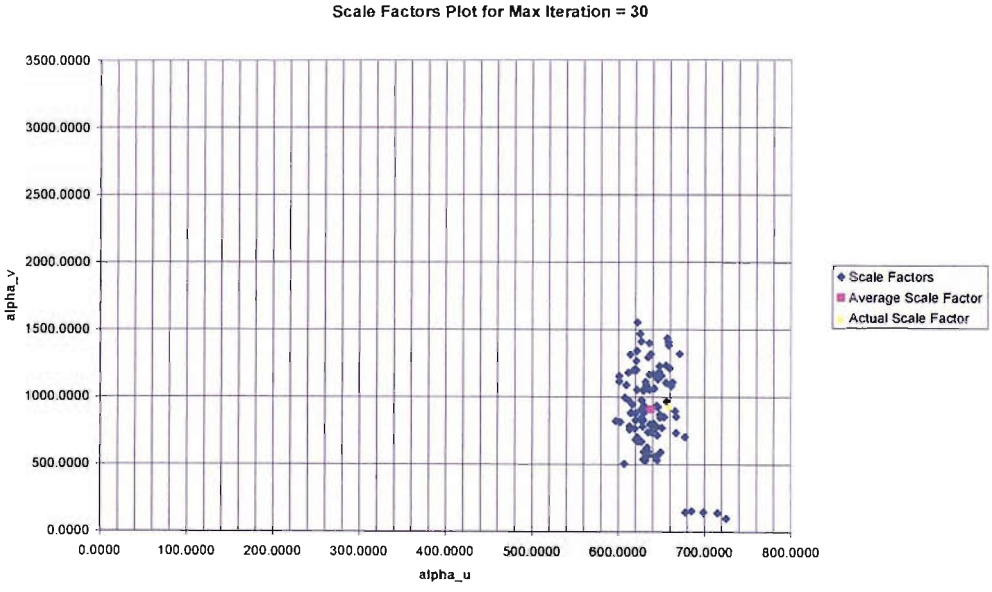


Figure 6.22: Scale Factors Plot where Maximum Iterations = 30.

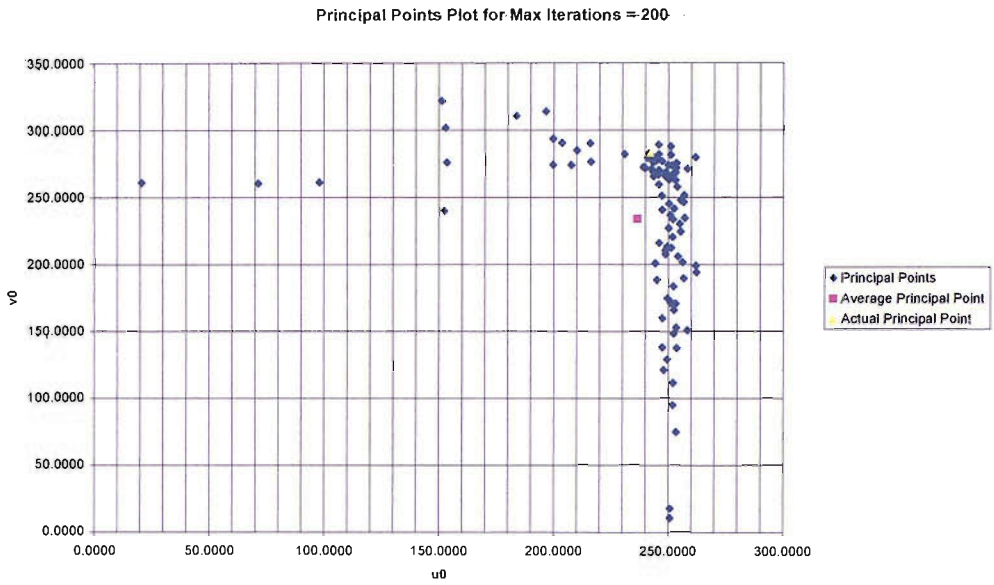


Figure 6.23: Principal Points Plot where Maximum Iterations = 200.

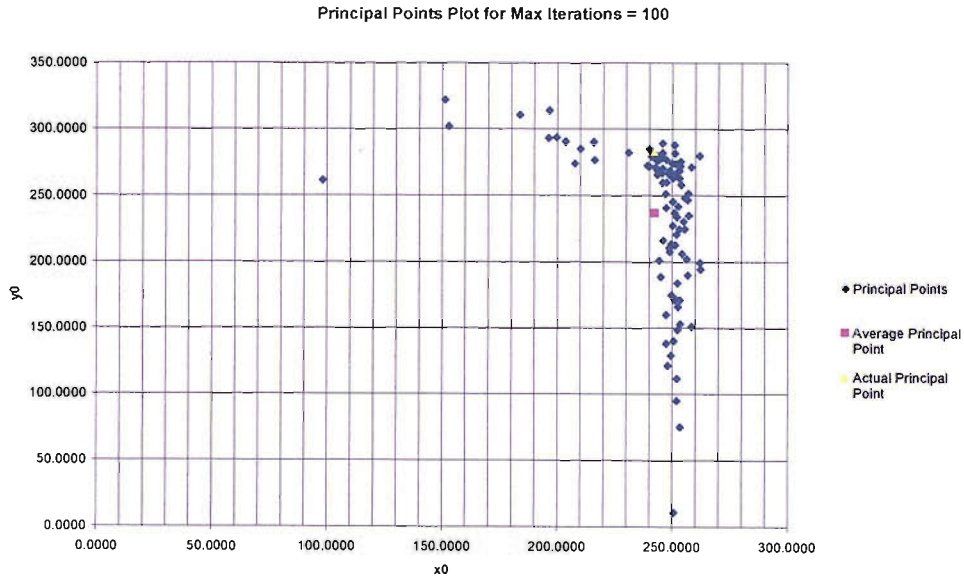


Figure 6.24: Principal Points Plot where Maximum Iterations = 100.

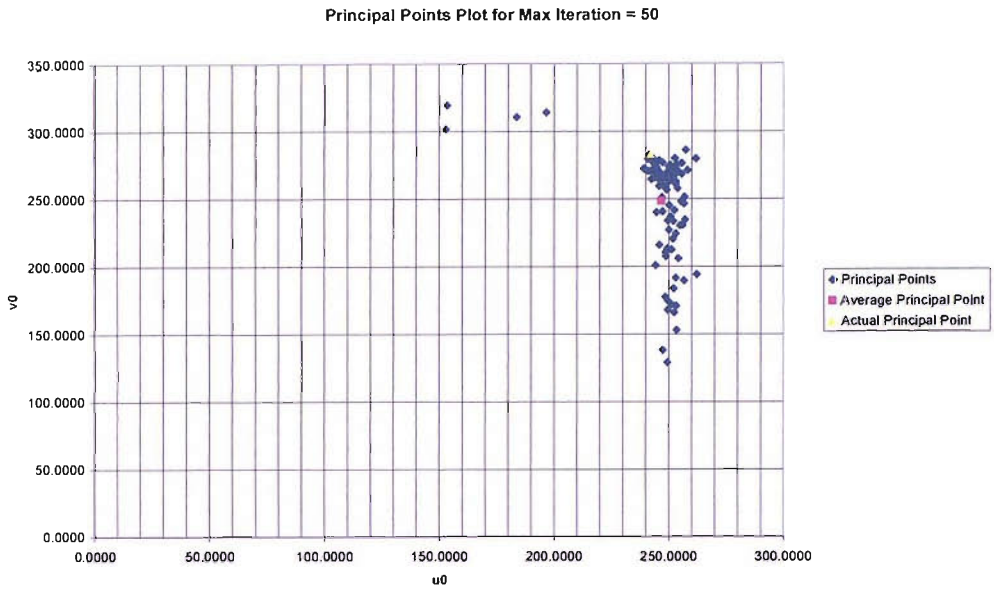


Figure 6.25: Principal Points Plot where Maximum Iterations = 50.

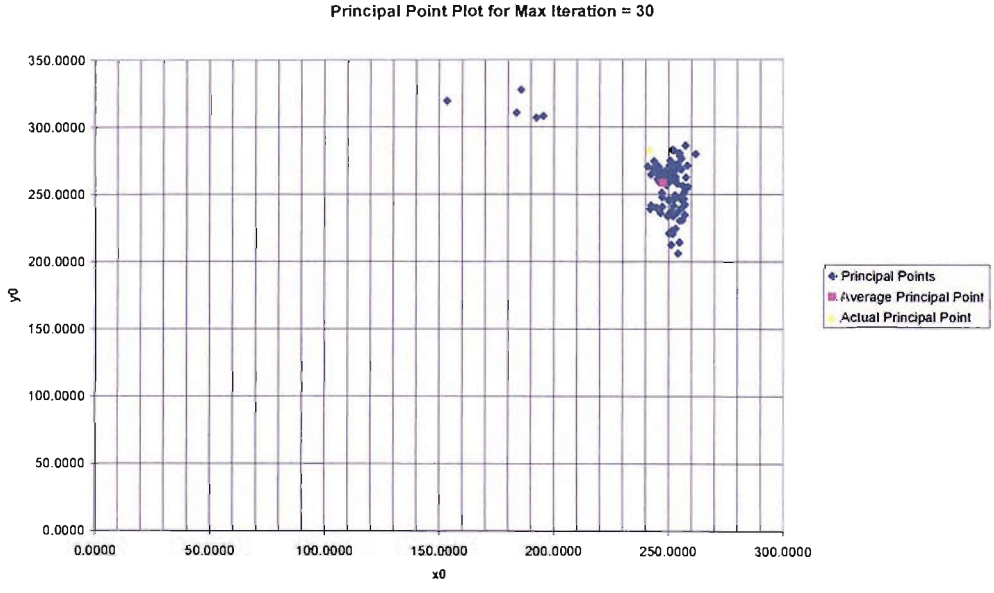


Figure 6.26: Principal Points Plot where Maximum Iterations = 30.

Table 6.8: Stability performance of SCAR when different initialisation values of α_u and α_v are used

Initialisation, $\alpha_u = \alpha_v$	α_u	α_v	u_0	v_0	Number of iterations
100	654.6860	927.0128	242.3709	271.2405	232
200	654.6860	927.0021	242.3709	271.2411	59
300	654.6860	927.0003	242.3709	271.2413	59
400	654.6860	926.9923	242.3708	271.2417	46
500	654.6848	926.9239	242.3705	271.2456	31
600	654.6853	926.9218	242.3705	271.2461	30
700	654.6858	926.9792	242.3708	271.2426	26
800	654.6863	927.0193	242.3710	271.2402	27
900	654.6859	926.9709	242.3708	271.2432	25
1000	654.6860	927.0003	242.3709	271.2412	26
1100	654.6860	927.0074	242.3709	271.2408	20
1200	654.6860	927.0110	242.3710	271.2405	24
1300	654.6861	927.0100	242.3709	271.2407	20
1400	654.6859	927.0041	242.3709	271.2410	22
1500	654.6863	927.0422	242.3712	271.2387	19
1600	654.6862	927.0381	242.3710	271.2390	21
1700	654.6861	927.0073	242.3710	271.2408	22
1800	654.6860	926.9955	242.3709	271.2416	24
1900	654.6861	927.0024	242.3709	271.2412	24
2000	654.6860	927.0009	242.3709	271.2412	24
2100	654.6858	926.9864	242.3708	271.2421	27

Table 6.9: Stability performance of SCAR when different initialisation values of u_0 and v_0 are used

Initialisation, $u_0 = v_0$	α_u	α_v	u_0	v_0	Number of iterations
0	654.686	927.002	242.3709	271.2412	24
100	654.6859	926.9945	242.3708	271.2416	28
200	654.686	926.998	242.3709	271.2414	22
300	654.686	926.9941	242.3709	271.2416	22
400	654.6861	927.0079	242.3709	271.2408	28
500	654.6872	927.174	242.3717	271.2306	27
600	654.6858	926.9808	242.3708	271.2424	30
700	654.6859	926.963	242.3708	271.2437	37
800	654.6877	927.2353	242.372	271.2269	34
900	654.6859	927.0007	242.3709	271.2411	30
1000	654.6861	927.0083	242.3709	271.2407	32
1100	654.6859	927.0049	242.3709	271.241	41
1200	654.6858	926.9905	242.3708	271.2419	39
1300	654.6861	927.0046	242.3709	271.241	42
1400	654.6858	926.9632	242.3708	271.2436	32
1500	654.6861	927.0069	242.3709	271.2407	59
1600	654.686	927.0006	242.3709	271.2413	31
1700	654.6861	927.0127	242.3709	271.2405	37
1800	654.686	926.9856	242.3709	271.2422	39
1900	654.686	926.9981	242.3708	271.2414	49
2000	654.6859	926.9925	242.3708	271.2418	121
2100	654.686	927.0001	242.3709	271.2413	66
2200	654.6861	926.9972	242.3709	271.2415	35
2300	654.6861	926.9992	242.3709	271.2413	65
2400	654.686	926.9939	242.3709	271.2417	40
2500	654.6859	926.9842	242.3708	271.2423	29

Table 6.10: Performance of SCAR in terms of speed for each algorithm.

Algorithm	*CPU Time (second)*	*Percentage (%)*
Corner detection	5.4	25.1
Point correspondence matching	3.4	15.7
Fundamental matrix estimation	7.7	35.6
Calibration matrix estimation	5.1	23.6
Total	**21.6**	**100**

Table 6.11: Computational complexity of SCAR system.

Stage	*Technique used*	*Computational Complexity*
Feature Detection	Harris corner detector	95 Addition and 22 Multiplication ops/pixel
	Harris refinement	37 Addition and 13 Multiplication ops/corner
Point Correspondence Matching	Correlation	$2W^2$ (Addition/Subtraction) and $4W$ (Multiplication) times B ops/corner
	Motion Vector Analysis	2N Subtraction
Fundamental Matrix Estimation	MAPSAC	Not fixed
Calibration Matrix Estimation	Algebraic Constraints	55 Addition and 27 Multiplication ops/iteration

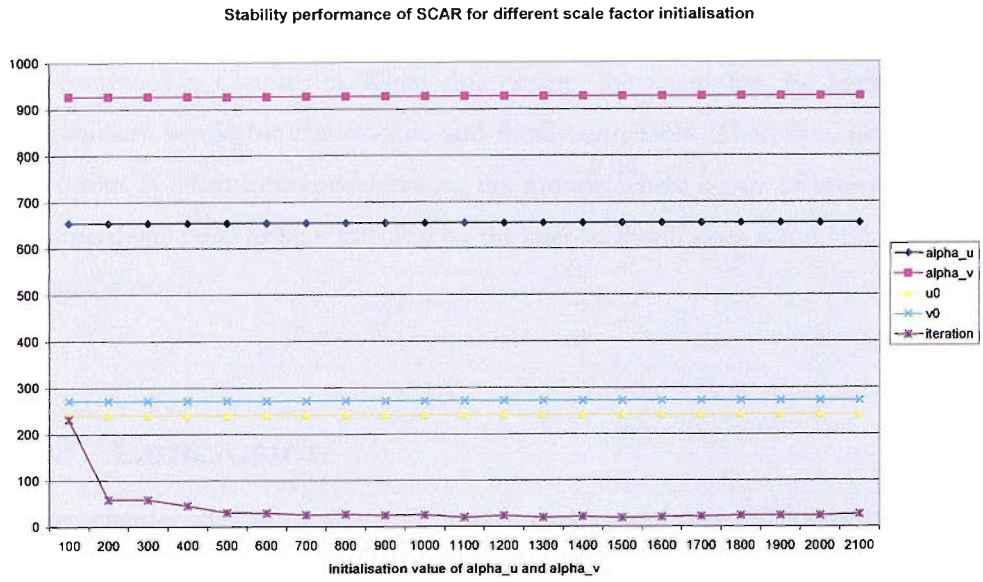


Figure 6.27: Stability performance of SCAR for different scale factor initialisation.

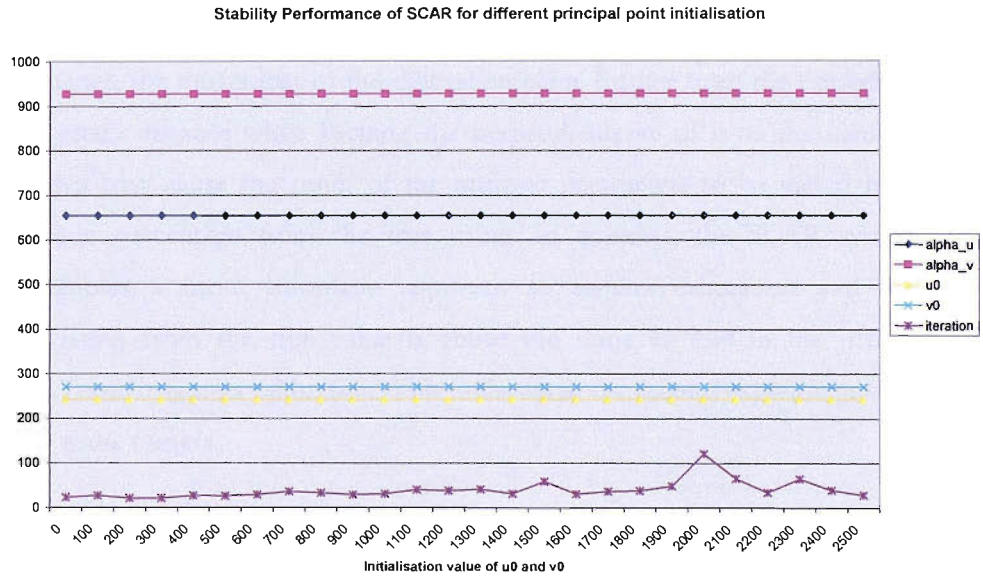


Figure 6.28: Stability performance of SCAR for different principal point initialisation.

A weakness that the SCAR system has is the inability to detect and discard pairs of images that undergo critical motion apart from case 1 as described in Section 5.5, Chapter 5. When this occurs, the result for the intrinsic parameters would be catastrophic and totally unreliable. Therefore, unless this issue is taken into consideration, the motion where a pair of images is captured will need to be controlled by the user so that it does not involve in critical motion.

6.5 Conclusion

This chapter describes the full implementation of the SCAR system developed throughout this thesis. It demonstrates the performance between the offline calibration in ARToolKit with the self-calibration in the SCAR system.

From the first experiment, the ARToolKit calibration procedures show the tedious steps that need to be followed every time a camera needs to be employed in an AR application. These steps involve user intervention, for instance, the movement of the calibration plane further from the camera by a certain distance while keeping the perpendicularity of it to the camera, which may cause the result of the intrinsic parameters to be varied by a certain percentage from the true value. In contrast, the SCAR system promotes a more automatic approach to camera calibration and the deviation from the true value is about the same as that in the offline ARToolKit camera calibration. This is shown in the second experiment with the static camera.

In the third experiment regarding images that come from a normal live video, the results show that, given a sequence of images that are noisy, the SCAR system is still able to obtain as reliable result as ARToolKit. When the distortion was corrected offline, the SCAR system showed

improvements compared with results without distortion correction. Therefore, if noisy and distorted images from live video are to be used in the SCAR system, the distortion needs to be corrected in order to obtain improved results for the camera parameters, especially when the distortion is severe.

The fourth and fifth experiments demonstrate the full implementation of the SCAR system, including the pre-calibration stages. Good results are obtained from both the Harris-SUSAN Hybrid corner detection and our point correspondence matching stage. These show that the SCAR system allows only good point matches to be passed through for the fundamental matrix estimation stage and calibration stage. In the fifth experiment, given these point matches as the input, the calibration stage shows robustness and stability of its intrinsic parameters results even though different initialisation values were chosen. The optimum number to be used for Maximum Iterations in Levernberg-Marquardt minimisation is only 30, which gives an advantage to the SCAR system in terms of speed.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Camera calibration remains an important topic in AR. The need to achieve high accuracy by using a known pattern, however, overlooks the need to provide a more flexible solution to calibrating a camera in AR. In the last few years many researchers have tried to address this issue from a number of different perspectives, focusing primarily on offline solutions for camera calibration in AR. This kind of solution makes the camera calibration a separate process from the AR system. An alternative approach, which is advocated in this thesis, is to integrate self-calibration into the AR system; this has the benefit of updating the intrinsic parameters online. This thesis has developed an AR system that incorporates self-calibration based on a moving camera. This chapter will summarise all the work presented in this thesis and recommend some future work.

The aim of this thesis has been to explore the integration of camera self-calibration in an AR system (SCAR). In particular, this research has been

concerned with updating the value of the intrinsic parameters of a monitor-based camera involved in an AR task. To achieve this, several pre-processing stages have been suggested, including a corner detection stage, a point correspondence matching stage, a fundamental matrix estimation stage and the self-calibration stage itself. Each chapter in this thesis describes the work that has been done in each of the aforementioned stages towards the completion of the camera self-calibration in AR.

In Chapter 1, it was stated that correct registration was a common problem in AR, and that good camera calibration was needed as one of the most effective solutions. In the available literature, most AR systems require a specified pattern to calibrate their cameras. We argued that even though the camera calibration in current AR systems that uses a plane-based solution provides good accuracy, it is less flexible, cumbersome and requires specific equipment setup. Furthermore, the intrinsic parameters of the camera cannot be corrected online if they change whether intentionally or unintentionally.

Having established the need for a more flexible approach to camera calibration, we analysed a typical AR system and proposed an alternative solution that replaces an *offline plane-based camera calibration AR system* with an *online integrated camera self-calibration AR system*. We decided that several stages need to be included as a pre-requisite to self-calibration, which consist of corner detection; point correspondence match and fundamental matrix estimation.

Having defined the proposed AR system, we introduced the general theory behind self-calibration and its derivation in Chapter 2, including our pinhole camera model, lens distortion and epipolar geometry.

Harris and SUSAN corner detectors have been proposed by Harris and Stephens [1988] and Smith and Brady [1995] respectively as methods of detecting high curvature features in an image. In Chapter 3, we reviewed these two corner detectors. The performance of each corner detector for stability, accuracy and speed was evaluated, and we discussed their applicability for point correspondence matching. We addressed the importance of accurate corners as input for the fundamental matrix estimation stage. Consequently, we proposed a Harris-SUSAN Hybrid corner detector in order to increase corner localisation. The search for true corners is performed by variable sizes of mask that changes adaptively with the number of connected region contained in two different sizes of mask window. Results show that localisation error is reduced by using the proposed corner detector.

Correlation functions are often used in searching for similarities in images or image parts. This method was reviewed in Chapter 4 to find point correspondences based on the detected corners method from Chapter 3. The consequence of applying correlation is a score value between -1 and 1. A corner point in one image is assumed to correspond with a point in another image when it has the highest correlation score. However, this is not always the case. A novel method based on motion analysis was developed which discards all outliers. It achieves this by employing two simple processes, mode of distance search and threshold setting. Results show that this method outperforms other techniques in terms of accuracy, simplicity and speed. A limitation of this method is that the performance is less efficient for forward or backward motion.

Chapter 5 starts by describing several techniques for self-calibration. In this thesis, an algebraic approach based on three views was used to solve for the intrinsic parameters of an AR camera. The presented method directly recovers the intrinsic parameters from the fundamental matrices and deals

with general camera motions. It has a simpler algorithm than other techniques and hence is suitable to be used for AR applications that require a fast algorithm in order to reduce lag in the system. In order to tackle the case when a camera with lens distortion is used, an algorithm to estimate distortion parameters was developed based on epipolar constraint. Based on the results obtained, distortion parameters can be estimated correctly when the distortion is less than 1.2×10^{-4} before they become unreliable. However, due to the length of time needed to estimate these parameters, the estimation of distortion parameters is better done offline.

Having finished developing the different stages, all of them were combined to build a complete AR system with built-in self-calibration. In Chapter 6 we demonstrated the performance of our AR system in comparison with ARToolKit, which uses offline camera calibration. We showed in our experiments that the SCAR system outperforms ARToolKit in terms of ease of use while maintaining the accuracy of the intrinsic parameters. We have already showed from the results that the pre-calibration stages have been designed in such a way that they only provide the best inputs for each stage. Provided that the camera motion is not critical, the SCAR system proves to be able to produce reliable intrinsic parameters. In order to achieve more consistent intrinsic parameters, the number of maximum iterations during calibration must be observed to a certain value. This condition, however, presents an advantage in terms of speed. The stability of the SCAR system, however, holds as long as it is not involved in any of the critical motion cases.

There are other things that need to be considered in order to make self-calibration a choice as an accurate and robust calibration method apart from plane-based calibration. The first consideration is the image-matching process in which the coordinates of the points between image pairs need to be correctly matched in order for the fundamental matrix to be correctly

estimated. A second consideration is the lighting conditions when capturing images. Since the image-matching algorithm is highly dependent on pixel correlation, different lighting conditions between views can degrade the ability to detect correct matches.

7.2 Future Work

A large part of the thesis has been concerned with the description of camera self-calibration in AR systems with all the necessary stages, which involve feature detection, point correspondence matching and fundamental matrix estimation. Whilst the development of the whole system has been shown to be successful, there is much scope for further improvement. In this section, some possible future recommendations based on the current work are presented.

The major limitation of using corner detectors in this work is that the system will not work efficiently when there are few corners available within the camera view. A more adaptive approach can be suggested to find other features such as curves, but at the expense of complexity.

Using motion analysis to detect outliers in point correspondence matching has limitations in the sense that outliers can hardly be detected when the camera is moving forward or backward. One possible improvement on the algorithm might include dividing the image into several segments. Then, independent motion analysis could be applied by assigning different mode values to each segment.

Feature detection and point correspondence matching can be improvised by the use of visual tracking to speed up the pre-calibration stages. The Lucas-Kanade algorithm may be employed to find the best features to track. However, problems may occur when there is occlusion in the scene. Further

research could be focused on the development of more flexible visual tracking for real-time correspondence matching.

In practice, a mechanism to detect and discard critical motion pairs from being processed by the system would be very important if we want to ensure the consistency of the intrinsic parameters results when performing the AR task. One suggestion for tackling the second case of critical motion is by employing the algorithm described in Mendonca [2001] that deals with a rotating camera where the principal axes of multiple camera positions intersect.

Bibliography

[Abdel-Aziz and Karara 1971] Y I Abdel-Aziz and H M Karara. Direct Linear Transformation Close-Range Photogrammetry. *Proc. ASP/UI Symp. Close-Range Photogrammetry,* pp. 1-18, January, 1971.

[Abdullah and Martinez 2002] J Abdullah and K Martinez. Camera Self-Calibration for the ARToolKit, in *The First IEEE International Augmented Reality ToolKit Workshop*, Darmstadt, Germany, September 2002.

[ARGOS] *ARGOS Virtual Pointer Camera Calibration Procedure.* WWW page = http://vered.rose.utoronto.ca/people/david_dir/POINTER/Calibration.html

[ARToolKit] *Augmented Reality Tool Kit - Human Interface Technology Lab.* WWW page = http://www.hitl.washington.edu/

[Asada and Brady 1986] H. Asada and M. Brady. The curvature primal sketch. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):2--14, 1986.

[Azarbayejani and Pentland 1995] A Azarbayejani and A P Pentland. Recursive estimation of motion, structure and focal length. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[Azuma 1993] R T Azuma. Tracking requirements for augmented reality. *Communications of the ACM,* Vol. 36, No. 7, pp. 50-51, July 1993.

[Azuma and Bishop 1994] R T Azuma and G Bishop. Improving static and dynamic registration in an optical see-through display. *Computer Graphics*, July 1994, pp. 194--204.

[Azuma 1997] R T Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, Vol. 6, No. 4, pp. 355-385, August 1997.

[Bajura 1993] M Bajura. Camera calibration for video see-through head-mounted display. UNC Chapel Hill Department of Computer Science Technical Report TR93-048 (July 7, 1993), 6 pages.

[Bajura et al. 1992] M Bajura, H Fuchs, and R Ohbuchi. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient, *Computer Graphics*, Vol. 26, No. 2, pp. 203-210, 1992.

[Ballard and Brown 1982] D H Ballard and C M Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1982.

[Bauckhage and Schmid 1996] C Bauckhage and C Schmid. Evaluation of keypoint detectors. Technical report, INRIA, 1996.

[Beardsley 1992] P A Beardsley. *Applications of Projective Geometry to Robot Vision.* PhD thesis, Department of Engineering Science, University of Oxford, Oxford, 1992.

[Berthilsson and Astrom 1997]. Reconstruction of 3D-curves from 2D-images using affine methods for curves. *In Proceedings of Computer Vision and Pattern Recognition*, 1997.

[Boufama and Mohr 1995] B Boufama and R Mohr. Epipole and fundamental matrix estimation using the virtual parallax property. *Proceedings of the 5th International Conference on Computer Vision*, pp. 1030-1036, Cambridge, Massachusetts, USA, June, 1995.

[Brannan et al. 1999] D A Brannan, M F Esplen, J J Gray. *Geometry,* Cambridge University Press.

[Brown 1971] D C Brown. Close-range camera calibration. *Photogrammetric Engineering*, Vol. 37, No. 8, pp. 855-866, 1971.

[Butterfield 1997] S Butterfield. Reconstruction of extended environments from image sequences. *PhD thesis*, School of Computer Studies, University of Leeds, Leeds, UK, 1997.

[Canny 1986] J F Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679--698, November 1986.

[Cazorla et al. 1999] M A Cazorla, F Escolano, D Gallardo, R Rizo. Bayesian Models for finding and Grouping Junctions. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 70-82, 1999.

[Cheok et al. 2002] A D Cheok, W Weihua, X Yang, S Prince, F S Wan, M Billinghurst, H Kato. *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 59-68, 2002.

[Coxeter 1969] H S M Coxeter. *Introduction to Geometry.* John Wiley and Sons, New York, 2nd edition, 1969.

[Coxeter 1974] H S M Coxeter. *Projective Geometry.* Springer-Verlag, New York, 1994 reprint of 2nd edition, 1974.

[Davis 1975] L S Davis. A Survey of Edge Detection Techniques. *Computer Graphics and Image Processing*, Vol. 4, pp. 248-270, 1975.

[Deriche 1987] R Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *Int. Journal of Computer Vision*, 1(2):167--187, 1987.

[Dornaika and Chung 2001] F Dornaika and R Chung. An algebraic approach to camera self-calibration. *Journal of Computer Vision and Image Understanding*, Vol. 83, No. 3, pp. 195-215, September, 2001.

[Drascic and Milgram 1991] D Drascic and P Milgram. Positioning accuracy of a virtual stereographic pointer in a real stereoscopic video world. *SPIE Proceedings* Volume 1457 – Stereoscopic Displays and Applications II, pp. 58-69, San Jose, California, September, 1991.

[Drascic et al 1993] D Drascic, J J Grodski, P Milgram, K Ruo, P Wong, and S Zhai. Argos: A display system for augmenting reality. *In Formal Video Programme and Proceedings of Conference on Human Factors in Computing Systems (INTERCHI'93)*, page 521, Amsterdam, Netherlands, 1993.

[Faugeras 1993] O D Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT Press, Cambridge, Massachusetts, USA, 1993.

[Faugeras 1995] O D Faugeras. Stratification of 3-D vision: projective, affine, and metric representations, *Journal of the Optical Society of America A,* Vol. 12, No. 3, pp. 465-484, March, 1995.

[Faugeras et al. 1992] O D Faugeras, Q T Luong and S J Maybank. Camera self-calibration: Theory and experiments. *International Proceeding of European Conference on Computer Vision,* pp. 321-334, Santa Margerita, Italy, 1992.

[Feiner et al. 1993] S. Feiner, B. MacIntyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53-62, July 1993.

[Figl et al. 2002] M Figl, W Birkfellner, C Ede, J Hummel, R Hanel, F Watzinger, F Wanschitz, R Ewers and H Bergmann. The Control Unit for a Head Mounted Operating Microscope Used for Augmented Reality Visualization in Computer Aided Surgery, *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 69-75, 2002.

[Fiorentino et al. 2002] M. Fiorentino, R. de Amicis, G. Monno, and A. Stork. Spacedesign: A mixed reality workspace for aesthetic industrial design. *In Proceedings of the IEEE and ACM ISMAR*, 86-94, 2002.

[Fishler and Boles 1981] M.A. Fishler and R.C. Boles. Random sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. Assoc. Comp, Mach., Vol 24, No 6, pp 381-395, 1981.

[Freeman and Davis 1977] H. Freeman and L.S. Davis. A corner finding algorithm for chain code curves. *IEEE Trans. on Computers*, 26:297-303, 1977.

[Fuhrmann et al. 2000] A Fuhrmann, D Schmalstieg, W Purgathofer. Practical calibration procedures for augmented reality. *Proceedings of the 6th Eurographics Workshop on Virtual Environments*, Amsterdam, Netherlands, June 2000.

[Gold et al. 1998] S Gold, A Rangarajan, C Lu, S Pappu and E Mjolsness. New algorithms for 2D and 3D point matching. *Pattern Recognition*, 31(8):1019-1031, 1998.

[Golub and Loan 1989] G H Golub and C F Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2nd Edition, 1989.

[Grafe et al. 2002] M Grafe, R Wortmann, H Westphal. AR-based Interactive Exploration of a Museum Exhibit. *The First IEEE International Augmented Reality Toolkit Workshop*, pp. 1-5, September, 2002.

[Grasset et al. 2001] R Grasset, X Decoret and J D Gascuel. Augmented reality collaborative environment: calibration and interactive scene editing, *Virtual Reality International Conference*, pp. 16-18, Laval Virtual, May, 2001.

[Grimson et al. 1994] WEL Grimson, GJ Ettinger, SJ White, T Lozano-Pérez, R Kikinis. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, pp. 430-436, Seattle, Washington, June, 1994.

[Grimson et al. 1995] W E L Grimson G J Ettinger, S J White, P L Gleason, T Lozano-Pérez, W M Wells III, R Kikinis. Evaluating and validating an automated registration system for enhanced reality visualization in surgery,

Proceedings of Conference on Computer Vision, Virtual Reality and Robotics in Medicine, pp. 3-12, Nice, France, April, 1995.

[Haralick 1984] R M Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(1):58-68, January, 1984.

[Haralick and Shapiro 1992] R M Haralick and L G Shapiro. *Computer and Robot Vision, Volume I.* Addison-Wesley, 1992.

[Harris and Stephens 1988] C G Harris and M Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 189-192, Manchester, 1988.

[Hartley 1992] R I Hartley. Estimation of relative camera positions for uncalibrated cameras. In G Sandini, ed., *Proc. 2nd European Conf. on Computer Vision,* Lecture Notes in Computer Science 588, pages 579-587, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[Hartley 1994] R I Hartley. Self-calibration from multiple views with a rotating camera. *Proceeding of the 3rd European Conference on Computer Vision*, Vol. 1, pp. 471-478, Stockholm, Sweden, May, 1994.

[Hartley 1994a] R I Hartley. Euclidean reconstruction from uncalibrated views. In J L Mundy, A Zisserman and D Forsyth, eds., *Applications of Invariance in Computer Vision*, Lecture Notes in Computer Science 825, pages 237-256. Springer-Verlag, 1994.

[Hartley 1995] R I Hartley. In defense of the 8-point algorithm. *Proceedings of the 5th International Conference on Computer Vision*, IEEE Computer Society Press, Boston, Massachusetts, pp. 1064-1070, 1995.

[Hartley and Zisserman 2000] R I Hartley and Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, Cambridge, UK, 2000.

[Holloway 1995] R Holloway. Registration errors in augmented reality. PhD dissertation. UNC Chapel Hill Department of Computer Science Technical Report TR95-016, August 1995.

[Holt and Netravali 1991] R J Holt, A N Netravali. Camera calibration problem: Some new results, *CVIU*, Vol. 54, No. 3, pp. 368-383, 1991.

[Horaud et al. 1995] R Horaud, R Mohr, F Dornaika, and B Boufama. The advantage of mounting a camera onto a robot arm, *In Proceedings of the Europe-China Workshop on Geometrical Modelling and Invariants for Computer Vision*, pp. 206-213, Xian, China, 1995.

[Huang and Netravali 1994] T Huang and A Netravali. Motion and structure from feature correspondences: A review, *Proceedings of IEEE,* Vol. 82, No. 2, pp. 252-268, February, 1994.

[Kutulakos and Vallino 1996] K N Kutulakos, and J R Vallino. Affine object representations for calibration-free augmented reality, *Proceedings of VRAIS '96* pp. 25-36, Santa Clara, California, 30 March - 3 April 1996.

[Kutulakos and Vallino 1998] K N Kutulakos and J R Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 1, January-March, 1998.

[Lenz and Tsai 1988] R K Lenz and R Y Tsai. Techniques for Calibration of the Scale Factor and Image Centre for High Accuracy 3D Machine Vision Metrology, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 5, pp. 713-720, September, 1988.

[Levenberg 1944] K Levenberg. A Method for the Solution of Certain Problems in Least Squares, *Quart. Applied Math*, Vol. 2, pp. 164-168.

[Livingston and State 1995] M Livingston and A State, Improved Registration for Augmented Reality Systems via Magnetic Tracker Calibration, University of North Carolina at Chapel Hill Technical Report TR95-037, 1995.

[Lorensen et al. 1993] W Lorensen, H Cline, C Nas, R Kikinis, D Altobelli, and L Gleason. Enhancing reality in the operating room. *In Proceedings of Visualization '93 Conference*, pp. 410-415, Los Alamitos, CA, October 1993. IEEE Computer Society Press.

[Lourakis and Deriche 2000] M A Lourakis and R Deriche. Camera self-calibration using the singular value decomposition of the fundamental matrix. In 4th Asian Conference on Computer Vision, volume I, pages 403-408, Taipei, Taiwan, January 2000.

[Lucas and Kanade 1981] B Lucas and T Kanade. An iterative image registration technique with an application to stereo vision. *In Proceedings of 7th International Joint Conference on Artificial Intelligence*, 1981.

[Luong and Faugeras 1997] Q T Luong and O Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices, *International Journal of Computer Vision*, Vol. 22, No. 3, pp. 261-289, 1997.

[Marquardt 1963] D Marquardt. An Algorithm for Least Squares Estimation of Nonlinear Parameters, *SIAM J. Applied Math,* Vol. 11, pp. 431-441, 1963.

[Marr and Hildreth 1980] D Marr and E Hildreth. Theory of edge detection. *Proceedings of Royal Society*, B 207:187-217, 1980.

[Marr and Hildreth 1991] D Marr and E Hildreth. Theory of edge detection. *In R Kasturi and R C Jain, editors, Computer Vision*, pp 77-107. IEEE, Los Alamitos, Ca, 1991.

[Maybank and Faugeras 1992] S J Maybank and O D Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision,* Vol. 8, No. 2, pp. 123-151, 1992.

[Medioni and Yasumoto 1987] G Medioni and Y Yasumoto. Corner detection and curve representation using cubic b-splines. *Computer Vision, Graphics and Image Processing*, 39:267--278, 1987.

[Mellor 1995a] J P Mellor. Enhanced reality visualization in a surgical environment. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, A.I. Technical Report No. 1544, January, 1995.

[Mellor 1995b] J P Mellor. Realtime camera calibration for enhanced reality visualization, *Proceedings of CVRMed `95,* pp. 471-475, Nice, France, April 3-5, 1995.

[Mendonca 2001] P R dos S Mendonca. *Multiview Geometry: Profiles and Self-Calibration*, PhD thesis, Department of Engineering Science, University of Oxford, Oxford, 2002.

[Mendonca and Cipolla 1999] P R S Mendonca and R Cipolla. A simple technique for self-calibration. In Proc. Conf. Computer Vision and Pattern Recognition, volume I, pages 500-505, Fort Collins, Colorado, June, 1999.

[Milgram and Kishino 1994] P. Milgram, F. A. Kishino. Taxonomy of Mixed Reality Visual Displays. *IECE Trans. On Information and System (Special Issue on Networked Reality)*, Vol. E77-D, No. 12. pp 1321-1329.

[Milgram et al. 1993] P. Milgram, S. Zhai, D. Drascic, and J.J. Grodski. Applications of augmented reality for human-robot communication. *In Proceedings of IROS '93: International Conference on Intelligent Robots and Systems*, pp 1467-1472, Yokohama, Japan, July, 1993.

[Moravec 1977] H P Moravec. Towards automatic visual obstacle avoidance. *In Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Carnegie-Mellon University, Pittsburgh, PA, August, 1977.

[Mundy and Zissermann 1992] J L Mundy and A Zissermann. *Geometric Invariance in Computer Vision.* Artificial Intelligence Series. MIT Press, Cambridge, USA, 1992.

[Ohta and Kanade 1985] Y Ohta and T Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transaction on PAMI*, 7(2):139-154, March, 1985.

[Pajdla and Hlaváč 1995] T Pajdla and V Hlaváč. Camera calibration and euclidean reconstruction from known translations, Presented at the workshop Computer Vision and Applied Geometry, Nordfjordeid, Norway, August 1-7, 1995.

[Pollefeys et al. 1998] M Pollefeys, R Khoch, and L Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. 6th Int. Conf on Computer Vision*, pages 90-95, Bombay, India, January, 1998.

[Pollefeys et al. 1999] M Pollefeys, R Koch and L Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters. Int Journal of Computer Vision, 32(1):7-25, August, 1999.

[Press et al. 1992] W H Press, S A Teukolsky, W T Vetterling, and B P Flannery. *Numerical Recipes in C*, Cambridge University Press, 2nd Edition, 1992.

[Prewit 1970] J M S Prewitt. Object enhancement and extraction. In B.S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*. Academic Press, 1970.

[Rangarajan et al. 1989] K Rangarajan, M Shah and D V Brackle. Optimal corner detector. *Computing Vision, Graphics and Image Processing*, Vol. 48, pp. 230-245, 1989.

[Roy and Cox 1997] S Roy and I Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proceedings of ICCV*, 1997.

[Salari and Sethi 1990] V Salri and I K Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transaction on PAMI*, 12(1):87-91, Jan 1990.

[Semple and Kneebone 1998] J G Semple and G T Kneebone. *Algebraic Projective Geometry*. Oxford Classic Texts in the Physical Sciences. Clarendon Press, Oxford, UK, 1998. Originally published in 1952.

[Shen and Castan 1992] J Shen and S Castan. An optimal linear operator for step edge detection. *Computer Vision, Graphics and Image Processing*, 54(2):112--133, March, 1992.

[Shen and Wang 2001] F Shen and H Wang. A local edge detector used for finding corners. *ICICS*, 2001.

[Shelton and Hedley 2002] B E Shelton and N R Hedley. Using Augmented Reality for Teaching Earth-Sun Relationships to Undergraduate Geography Students. *The First IEEE International Augmented Reality Toolkit Workshop*, pp. 6-13, September 2002.

[Slama 1980] C C Slama, editor. *Manual of Photogrammetry,* American Society of Photogrammetry, fourth edition, 1980.

[Smith and Brady 1995] S M Smith and J M Brady. SUSAN - A New Approach to Low Level Image Processing. *Technical Report TR95SMS1c*, Oxford University, 1995.

[Springer 1964] C E Springer. *Geometry and analysis of projective spaces.* Freeman, San Francisco, USA, 1964.

[Sobel 1990] I Sobel. An isotropic 3x3 image gradient operator. *In H. Freeman, editor, Machine Vision for Three-Dimensional Scenes*, pages 376--379. Academic Press, 1990.

[Sojka 2003] E Sojka, *A new approach to detecting the corners in digital images*, IEEE ICIP 2003.

[Sonka et al. 1999] M Sonka, V Hlavac, R Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd Edition, PWS, 1999.

[State 1995] M L State, Improved Registration for Augmented Reality Systems via Magnetic Tracker Calibration. University of North Carolina at Chapel Hill Technical Report TR95-037, 1995.

[State et al. 1996] A State, M A Livingston, W F Garrett, G Hirota, M C Whitton, E D Pisano, and H Fuchs. Technologies for augmented reality systems: Realizing ultrasound-guided needle biopsies, *Proc. SIGGRAPH '96*, pp. 439-446, 1996.

[Sturm 1997] P Sturm. Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction. *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 1100-1105, San Juan, Puerto Rico, June, 1997.

[Sturm 2002] P Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length, *Image and Vision Computing*, volume 20, pp. 415-426, 2002.

[Sturm and Maybank 1999] P F Sturm and S J Maybank, On plane-based camera calibration: A general algorithm, singularities, applications, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, pp. 432-437, 1999.

[Sudhir et al. 1997] G Sudhir, S Banerjee and A Zisserman. Finding point correspondences in motion sequences preserving affine structure. *Computer Vision and Image Understanding*, 68(2):237-246, November 1997.

[Tordoff and Murray 2000] B Tordoff and D W Murray, Violating rotating camera geometry: The effect of radial distortion on self-calibration, *International Conference on Pattern Recognition*, 2000.

[Torr 1995] P Torr. *Motion Segmentation and Outlier Detection.* PhD thesis, Department of Engineering Science, Oxford, 1995.

[Torr 2002] P H S Torr, A structure and motion toolkit in Matlab "Interactive adventures in S and M", Technical Report MSR-TR-2002-56, June, 2002.

[Torr 2002a] P H S Torr, Bayesian Model Estimation and Selection for Epipolar Geometry and Generic Manifold Fitting, *Int. Journal on Computer Vision*, 2002.

[Triggs 1996] Factorisation methods for projective structure and motion. *In Proc. Conf. Computer Vision and Pattern Recognition*, pages 845-851, San Francisco, USA, 1996.

[Triggs 1997] B Triggs. Autocalibration and the absolute quadric. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 609-614, San Juan, Puerto Rico, June, 1997.

[Triggs 1998] B Triggs, Autocalibration from planar scenes, *ECCV*, pp. 89-105, 1998.

[Trivedi 1988] H P Trivedi. Can multiple views make up for lack of camera registration? Image and Vision Computing, 6(1):29-32, February, 1988.

[Tsai 1985] R Y Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. Research Report RC 11413 (#51342), IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, September, 1985.

[Tsai 1987] R Y Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automotion,* Vol. RA-3, No. 4, August, 1987.

[Tuceryan et al. 1995] M Tuceryan, D S Greer, R T Whitaker, D Breen, C Crampton, E Rose, and K H Ahlers. Calibration Requirements and

Procedures for Augmented Reality, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 3, pp. 255-273, September, 1995.

[Uenohara and Kanade 1995] M Uenohara and T Kanade. Vision-based object registration for real-time image overlay, *Proc. CVRMED '95*, pp. 14-22, 1995.

[Vallino 2002] J Vallino. Introduction to Augmented Reality, www.se.rit.edu/~jrv/research/ar/introduction.html, page last modified 22 August 2002.

[Wang and Brady 1995] H Wang and M Brady. Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, Vol. 13:9, pp. 695-703, 1995.

[Wei and Ma 1994] G Wei and S Ma. Implicit and explicit camera calibration: Theory and experiments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 5, pp. 469-480, 1994.

[Wellner 1993] P Wellner. Interacting with paper on the digital desk. *Communications of the ACM*, 36(7):87-96, July 1993.

[Whitaker 1995] R T Whitaker, C Crampton, D E Breen, M Tuceryan and E Rose. Object Calibration for Augmented Reality. *Proceedings of Eurographics '95,* pp. 15-27, Maastricht, The Netherlands, August, 1995.

[Zeller and Faugeras 1996] C Zeller and Faugeras. Camera self-calibration from video sequence: The Kruppa equations revisited. *Technical Report RR-2793*, INRIA, France, February, 1996.

[Zhang et al. 1994] Z Zhang, R Deriche, O Faugeras, Q T Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *INRIA Research Report* No. 2273, May 1994.

[Zhang 1996] Z Zhang. On the epipolar geometry between two images with lens distortion, *Proceeding of International Conference on Pattern Recognition*, Vol. 1, pp. 407-411, Vienna, August, 1996.

[Zhang 1998] Z Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, Vol. 27, No. 2, pp. 161-198, 1998.

[Zisserman et al. 1998] A Zisserman, D Liebowitz and M Armstrong. Resolving ambiguities in auto-calibration. *Phil. Trans. Royal Soc. London A*, 356: pp. 1193-1211, 1998.

[Zuniga and Haralick 1983] O A Zuniga and R M Haralick, Corner detection using facet model, *Proc. Conference on Pattern Recognition and Image Processing*, pp. 30-37, 1983.

Appendix A

MARKER PATTERN FOR THE ARTOOLKIT

The marker pattern is designed in such a way that it can be detected and tracked as easy as possible. The followings are example of marker pattern used by the ARToolKit (refer Figure A.1 and A.2). Note that the marker pattern consists of two separate patterns:

1) Outside pattern which is a thick black square.
2) Inside pattern which can be in different shape. The users can have the chance to design their own inside pattern to suit their application.

Therefore, pattern detection process in ARToolKit involves two stages; detecting the outside pattern and subsequent to this, the system will search for the inside pattern.

As a rule, one inside pattern corresponds to only one virtual object (refer Figure A.3 (a) and A.3 (b)). Therefore when the computer detects a particular pattern, the corresponding virtual object will be seen overlaid on

the pattern. If the camera has been calibrated, the virtual object will be overlaid according to the position and orientation of the pattern (refer Figure A.3 (c)). The OpenGL API is used for setting the virtual camera co-ordinates and drawing the virtual images.

Figure A.1: Hiro pattern

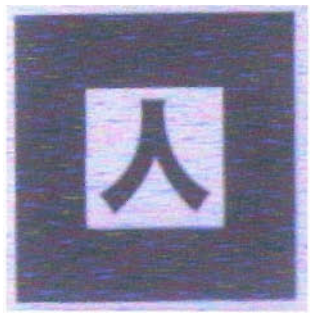
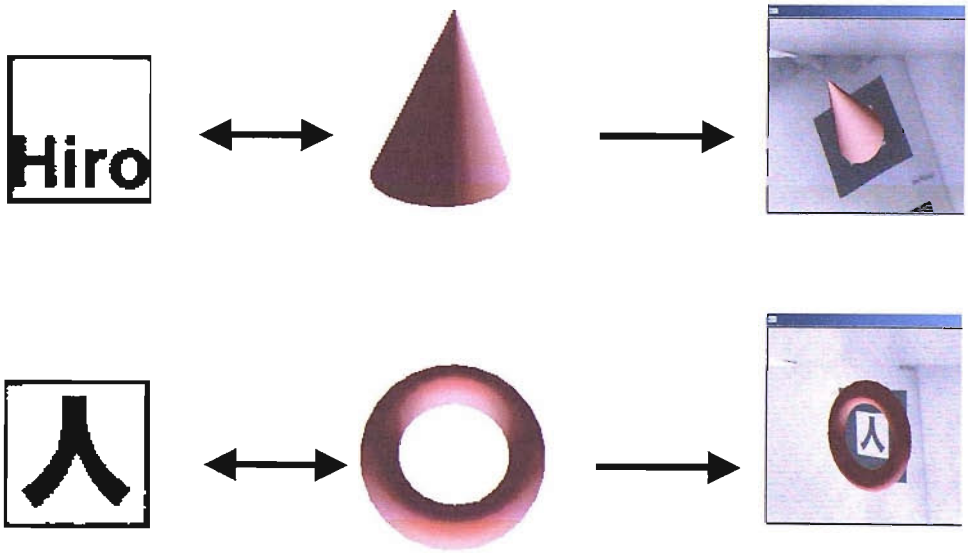


Figure A.2: Kanji pattern



(a) (b) (c)

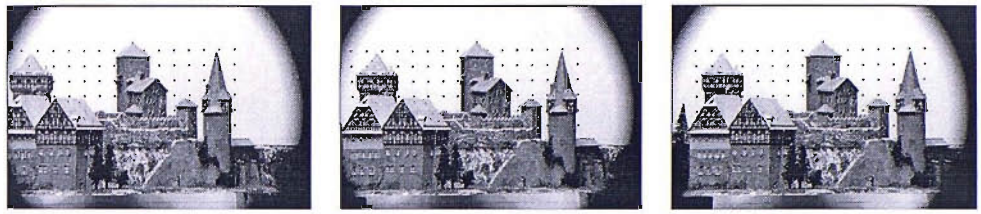
Figure A.3: Augmenting patterns with virtual objects.

Appendix B

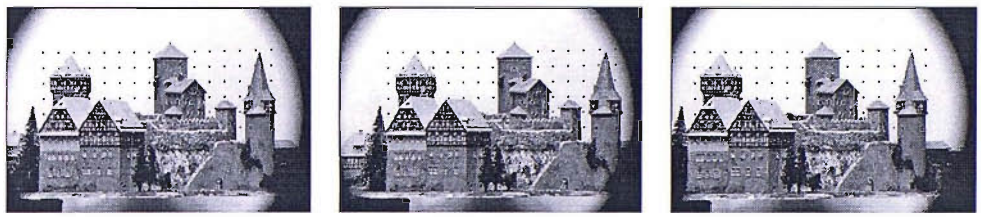
TEST IMAGES

toto1 toto2 toto3

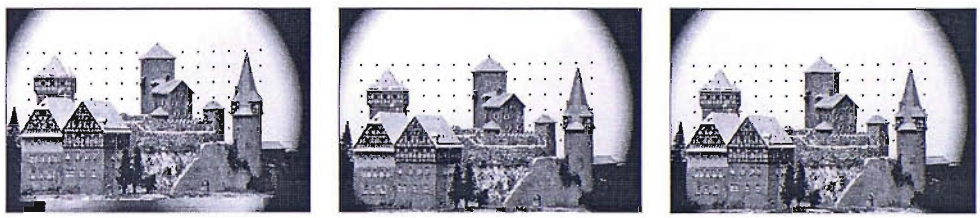
Figure B.1: Lab sequence



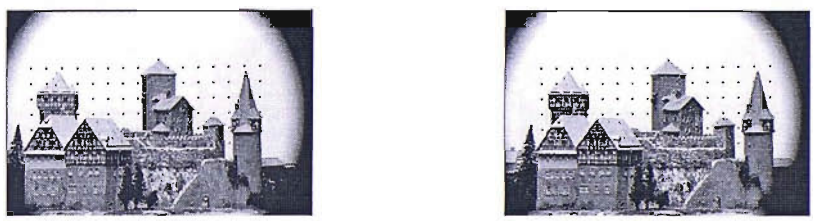
bighouse_frame_000 bighouse_frame_001 bighouse_frame_002



bighouse_frame_003 bighouse_frame_004 bighouse_frame_005



bighouse_frame_006 bighouse_frame_007 bighouse_frame_008



bighouse_frame_009 bighouse_frame_010

Figure B.2: House sequence

Figure B3: Fountain sequence