Retrieving Camera Parameters from Real Video Images

by

Ning Li

B. Sc., Xidian University, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

Department of Computer Science

We accept this thesis as conforming

to the required standard

The University of British Columbia

August 1998

Abstract

The motivation behind Computer Augmented Reality (CAR) is to effectively merge real video images and computer generated images to enhance the usefulness of the two sources of information. To fulfill this goal, we need to solve many related tasks which need expertise both in computer graphics and computer vision. This thesis is focused on one important aspect of CAR: establishing common viewing conditions between real video and computer generated images, that is, recovering the camera extrinsic parameters (position and orientation) as well as camera intrinsic parameters from real video images, so that the computer generated objects can be inserted accordingly, and the combined environment can be similarly rendered.

In this thesis, we assume that the camera is moving when the real video images are taken, and we know nothing about the camera intrinsic parameters - the camera is uncalibrated. First a corner detector is used to acquire some feature points in the image. With the knowledge of Euclidean 3D measurements for six or more feature points in the image, the matrix which transforms 3D coordinates into 2D coordinates can be acquired fairly accurately, then the camera parameters can be recovered from this matrix. A system is built which makes the whole process work as automatically as possible.

We tested our system with both synthetic and real images. We found that the system is able to produce useful results in most cases. When applied in the context of CAR to insert computer generated objects into real images, the results provide useful information about the real camera, so the synthetic camera can be manipulated accordingly.

Contents

List of Figures

Acknowledgments

First of all, I would like to express my heartfelt thanks towards my supervisor, Dr. Alain Fournier, for his guidance, invaluable support, and extreme patience through my research. His suggestions and advice have provided me with knowledge and experience beyond the scope of my thesis. I would also like to thank my second reader, Dr. David Lowe for reading through my thesis, and gave valuable advice.

Many thanks to friends, both inside and outside UBC, who have helped me in some very difficulty times. They have made it possible for me to have a joyful life during my stay at UBC.

Ning Li

The University of British Columbia

August 1998

Chapter 1

Introduction

1.1 Computer Augmented Reality

In the past twenty years, computer graphics has made great strides towards producing realistic images. Improved hardware and software has led to increased realism in modeling shape and rendering lighting effects. But neither the hardware nor the software has developed to the level of producing realistic images of our everyday environment in real time. The use of real video images eliminates the need to model complex environments in great detail, and, by nature, provides a realistic image to the user. On the other hand, real video images are not sufficient for many applications; sometimes it is necessary to insert computer-modeled objects into a real existing environment, or insert real objects into a computer modeled environment.

Computer Augmented Reality (CAR) is the process by which real images can be enhanced by superimposing additional, usually computer-generated, visual objects. In the words of Alain Fournier [Fournier 93]: “The usefulness of the two sources of information, real video images (RVI) and computer generated images (CGI) can only be enhanced by the ability to merge them freely in real time on the workstation screen. By merging we mean ideally in a way that appears “seamless” where one cannot distinguish between the “real” part and the “synthesized” part. We call the ensemble of techniques to reach that goal Computer Augmented Reality (CAR)”.

Computer Augmented Reality can be found in a wide range of applications such as visualization (e.g. medical imagery, architecture design), and video production (movie special effects, advertising, impact studies). In most cases, the main problem is to ensure the accuracy of the superimposition between the real images and the synthetic one. When mixing 3D synthetic and real objects in a same animated sequence for video applications, one must insure the light and shadow consistency as well as the detection of the parts of the virtual objects hidden by the real ones in the resulting images. This task poses many challenges, and they can be partitioned into several groups of “determining scene geometry, establishing common viewing conditions, and establishing common illumination in addition to rendering” [Fournier 93].

Determining the scene geometry involves obtaining geometric models of the real objects and determining the mutual visibility of the real and synthetic objects while com-

positing the two images. With full 3D information, the relative positions and occlusion between synthetic and real objects in the final images can be obtained. Errors in the 3D models will affect the calculation of the visibility and the illumination later on.

Methods to establish common viewing conditions can be distinguished as active methods and passive methods. In active methods, the real camera is controlled and/or monitored to give the relevant data. In passive methods the internal (focal length, piercing point) parameters and the external parameters (position and orientation) of the camera are retrieved from the RVI and are used to set the synthetic camera parameters in CGI accordingly, so that the combined environment can be similarly rendered. If the real values and those used in rendering are inconsistent, we will perceive the misalignment, different perspective and motion of stationary objects in the final images.

Establishing common illumination includes computing both local illumination and global illumination of RVI from computer generated light sources, and those of CGI from real light sources. This is necessary for shading the synthetic objects so that they appear to be illuminated by the real lights or for shading the real objects according to the CGI lights. Common illumination requires full 3D information, and should use explicit modeling of the real world objects. If this is not properly done, then shading inconsistencies may appear, we can observe that one component of the objects may be too light or dark, or fail to cast certain shadows, and shadows are too light or dark.

1.2 Motivation and Goal

This thesis, part of a research effort of the Computer Augmented Reality (CAR), focuses on the problem of establishing common viewing conditions between real and computer generated images. The specific goal of this project is to determine the camera movement as well as the camera intrinsic parameters from the real video images using a calibration free method, so we can insert computer generated objects according to the recovered parameters.

Much work has been done in computer vision on this topic, but usually these steps are followed:

- calibrating the camera and getting the camera intrinsic parameters;
- assuming a priori knowledge of the 3D geometry of objects, or performing 3D reconstruction;
- sometimes assuming some previous knowledge about the camera motion.

However, these may not be appropriate for the CAR context. For example, assuming that camera is calibrated is not realistic for a real time CAR situation. First, the calibration is difficult to obtain and very sensitive to errors. Second, for many applications it is not possible to calibrate on-line, for example a calibration pattern is not available or the camera is involved in other visual tasks. Third, the images can be taken by different cameras or by a single camera whose focus length is changing at different times, so the camera

intrinsic parameters are variable. Also for some objects, it is difficult to know fully their 3D geometry, but it is relatively easier to measure a few points on them especially when these points are corners.

In this thesis, we describe an approach to video-based CAR that avoids the camera's calibration; it does not use any metric information about the calibration parameters of the camera; it does not know the 3D geometry of objects; it does not involves 3D reconstruction, and there is no limitation in principle to the camera motion while capturing the images. The only requirement is the ability to track across frames at least 6 points acquired by feature detection whose world coordinates are known. Our goal is to design a system in which the determination of the camera parameters can be done as automatically as possible. The ultimate goal is to achieve real-time update of the parameters.

We have to make some assumptions about our system in order to work properly and to get the best results:

- There are enough corners in the real scene, i.e. the bounding contours of the objects cannot all consist of smooth curves;
- The objects in the scene where corners are found are rigid, and none of them is moving, so the displacements of these objects are caused by the camera moment;
- The camera used is such that non affine or non perspective distortion in the images can be ignored;

- The coordinates of some corners in the world coordinate system can be measured accurately in advance.

1.3 System Design

A diagram of the system is shown in Figure 1.1, and can be summarized as:

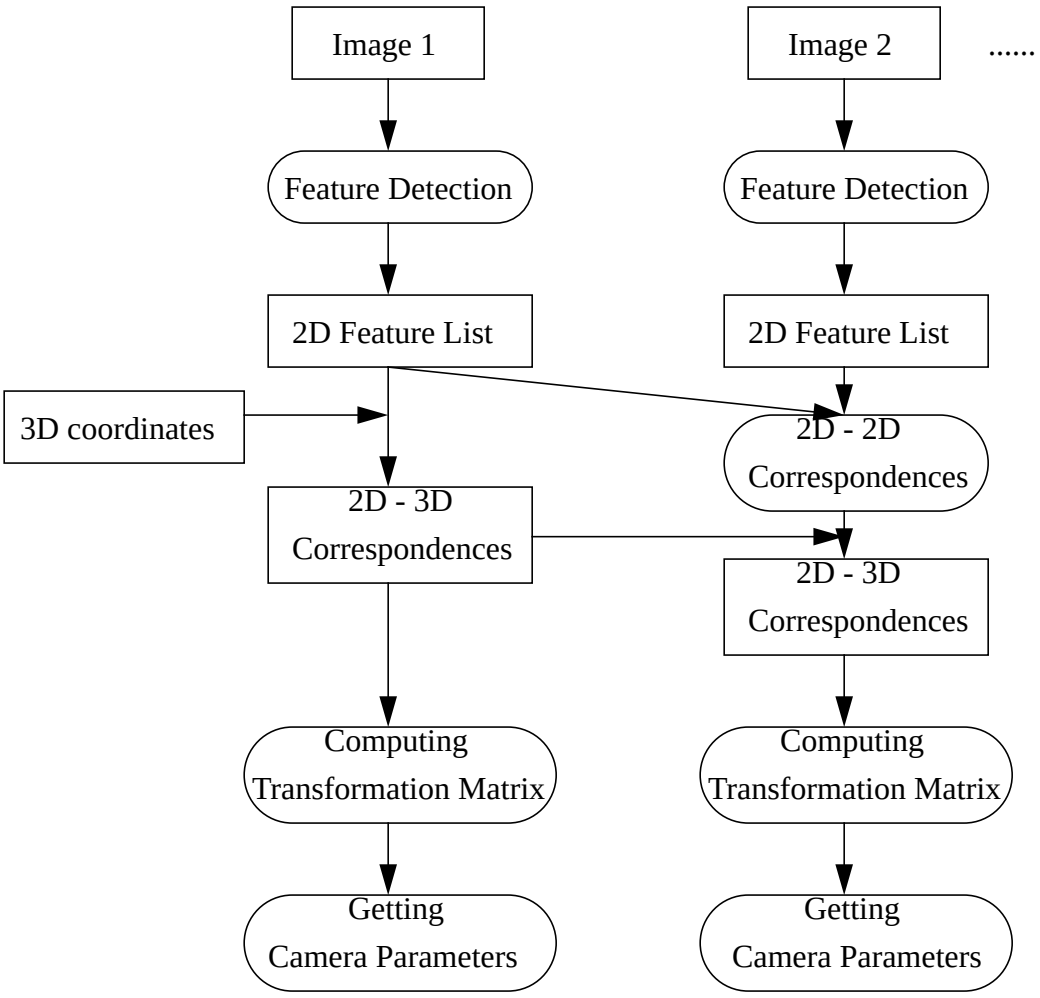

Figure 1.1 System Design

Feature Detection: The strongest features (corners in this case) are found in each image by a corner detection algorithm and output to a feature list for further use. In our work the Harris corner detection algorithm will be used.

2D-3D Feature Correspondences: For the first image, we get 2D image coordinates for the feature points using a corner detection algorithm. With the imported 3D world coordinates for some feature points, we can match these 2D points in the image coordinates to 3D points in the world coordinate. By doing this, we get the 2D-3D feature correspondences list. Note that this kind of 2D-3D point matching is done only for the first image, or for the images in which the number of points with 3D coordinates are less than 6.

2D-2D Feature Correspondences: For the following images, after getting the 2D image feature points for each image, the system will automatically match these feature points to the ones in the previous image to get a 2D-2D feature correspondence list. With the previous 2D - 3D matching information, the 3D world coordinates can be automatically transferred to some of the 2D feature points in this image. A new 2D-3D point correspondence list is then obtained for each image.

From feature correspondences to the transformation matrix: With the 2D-3D point correspondences list for each image, the relationship between the 3D coordinates of a point and the corresponding 2D coordinates of its image can be expressed in terms of 3 by 4 matrix using the homogeneous coordinate system. This matrix is generally known as the transformation matrix. It maps a 3D point, expressed with respect to the world coordi-

nates, onto its 2D image whose coordinates are expressed in viewpoint units. Here we assume that a pin-hole camera model in used, so that projective geometry can be used. Such a matrix can be determined experimentally by measuring the image coordinates of 6 or more points whose 3D coordinates are known. This matrix contains all the geometric information about of the imaging process which is possible to obtain from uncalibrated image.

From the transformation matrix to camera parameters: The last step is the determination of the camera parameters. Ganapathy's method [Ganapathy 84] will be used in our system. In this method, the transformation matrix is represented in a way different from traditional viewing matrix used in computer graphics. It is expressed using 10 camera parameters (6 extrinsic and 4 intrinsic). By putting some other constraints, it is possible to derive camera intrinsic parameters and extrinsic parameters analytically from the transformation matrix without resorting to non-linear methods.

1.4 Thesis Structure

Chapter 2 provides some background about projective geometry, the camera model we will use in this system and camera parameters relevant to this research. Chapter 3 summarizes several corner detection methods, and analyzes the method we use in our system in detail. Chapter 4 analyzes the matching method we used to match feature points. Chapter 5 deals with the decomposition method we used to get the camera intrinsic and extrinsic parameters from the projective transformation matrix. Chapter 6 presents the experimental

results for some real images and some synthetic images. Chapter 7 gives some ideas for future work and the overall conclusions.

Chapter 2

Background

This chapter presents background material on projective geometry, camera models and camera parameters.

2.1 Projective Geometry

The image formation process for realistic rendering is comprised of two components, geometric and radiometric. the geometric component determines where a surface element in the scene appears in the image and the radiometric component determines the image brightness of the surface element. The primary focus of this thesis is on the geometric component of the image formation process. Projective geometry deals with the general case of perspective projection and therefore provides clear understanding of the geometric aspects of image formation.

We will present a short introduction to the definitions and vocabulary of projective geometry. The reader is referred to [Mohr 96] for a general introduction or to [Semple 52] for advanced vision oriented consideration on projective geometry.

Projective Space Given a coordinate system, n-dimensional real *affine space* is the set of points parameterized by the set of all n-component real vectors $(x_1, \ldots, x_n) \in R^n$. Similarly, the points of n-dimensional *projective space* $(x_1, \ldots, x_{n+1}) \in R^{n+1}$, with the conditions that at least one coordinate must be non-zero and that the vectors are equivalent: $(x_1, \ldots, x_{n+1}) \sim \lambda (x_1, \ldots, x_{n+1})$ (2.1)

They represent the same point of P^n for all $\lambda \neq 0$. The x_i are called the *homogeneous coordinates* for the projective point.

The usual n dimensional affine space R^n is mapped into P^n through the correspondence Ψ:

$$\Psi: \quad (x_1, \ldots, x_n) \rightarrow (x_1, \ldots, x_n, 1) \tag{2.2}$$

Ψ is a one to one mapping, it provides us with an understanding of the points $(x_1, \ldots, x_n, 1)$ which can be viewed as the usual point in the Euclidean space.

A **projective transformation** from P^n into P^k is the mapping defined by a $(k+1) \times (n+1)$ full rank matrix W such that the image of $(y_1, \ldots, y_{k+1})$ is defined in the usual way, represented in homogeneous coordinates:

$$\lambda \begin{bmatrix} y_1 \\ \dots \\ y_{k+1} \end{bmatrix} = T \cdot \begin{bmatrix} x_1 \\ \dots \\ x_{n+1} \end{bmatrix}$$

Since the column vectors are defined up to a scaling factor, the matrix T is too. Therefore, the matrix has $(k+1) \times (n+1) - 1$ degrees of freedom.

A particular case is the perspective projection which maps the 3D space P^3 into the image plane P^2. Expressed in homogeneous coordinates, T is a 3×4 matrix, and has 11 degrees of freedom. Through this paper, we use the terms *the transformation matrix* and *the projection matrix* interchangeably to refer to the matrix T.

2.2 Camera Model

The geometry of the image formation process can be represented by a set of canonical camera models that describe the projection of a scene onto the image plane. The classic model for a camera is a pinhole at a fixed distance from an image plane. In first approximation light travels in a straight line, hence, each point in the image specifies a ray which extends towards the scene. This gives us the standard perspective projection model which is only an approximation to the optical physics of a physical camera - an excellent approximation in most applications. The other models are specializations of this model. [Shapiro 90] discusses six camera models which are in two classes, calibrated and uncalibrated. The projective and affine camera models are in the uncalibrated class. As we only use

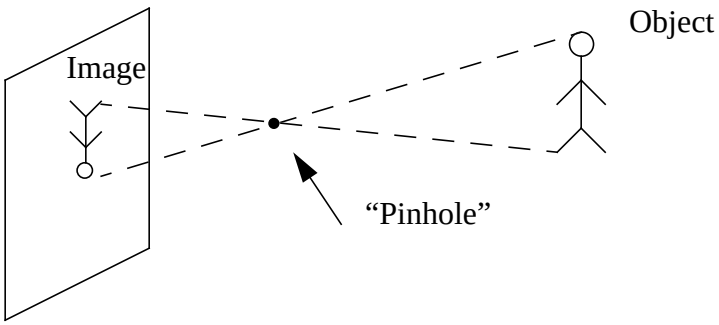

Figure 2.1 Pinhole Camera Model

uncalibrated camera in this research, we only discuss the projective camera model, i.e. the pinhole camera model.

2.1.1 Projective Camera

Again the camera model most widely used in practice is the pinhole (projective camera) model illustrated in Figure 2.1. Ideally, a pinhole camera has an infinitesimally small aperture, through which light enters the camera and forms an image on the surface facing the aperture, and the camera performs a perspective transformation of 3D space to the two dimensional image plane. The basic assumption behind this model is that the relation between the world coordinate and the image coordinate is linear projective, i.e., straight lines project to straight lines, and consequently this model does not take into account lens distortion or image plane defects but allows us to use the powerful tools of projective geometry. Sometimes this model has to be corrected for distortion, and if a very high accuracy is needed more sophisticated correction methods are necessary.

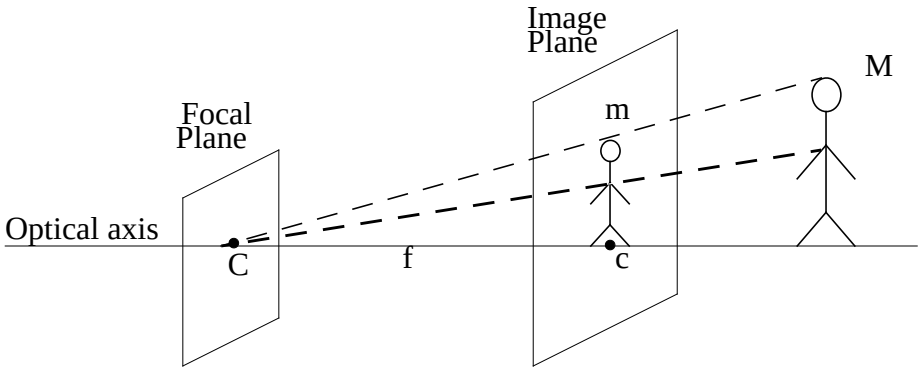

Figure 2.2 Perspective Projection

As illustrated in Figure 2.1, the image in a pinhole model is inverted, this inversion of the image is inconvenient as far as analysis goes. Hence, it is customary to consider the geometric model of Figure 2.2 equivalent to Figure 2.1, in which the image is on the same side of pinhole as object, as a result, the image is not inverted. It consists of a plane R called the image plane in which the image is formed through a perspective projection: a point C, the optical center, located at a distance f, the focal length of the optical system, is used to form the image m in the image plane of the 3D point M as the intersection of the line $\langle M, C \rangle$ with the image plane. The optical axis is the line going through the optical center C and perpendicular to image plane, which it pierces at a point c. Another plane of interest is the focal plane going through C and parallel to image plane.

It is convenient, from a mathematical point of view, to consider the world as embedded in a three dimensional projective space, P^3 and the image plane as embedded in a

projective space of dimension two, P^2. This facilitates the expression of the projective transformation. Points in the scene and image projective spaces are represented as vectors in homogeneous coordinates. The projective transformation from P^3 to P^2 then is given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [T] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [M]\ [N] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.3}$$

when T is a 3×4 matrix, which maps the scene point $p = [x, y, z]^T$ to image point $w = [u, v]^T$ expressed in homogenous coordinates. M is a 3 by 4 matrix, N is a 4 by 4 matrix, the product of these two matrix equals to T, the Transformation Matrix.

2.3 Camera Parameters

The purpose of this section is to describe what the matrices M and N should be when we relate 3D world coordinates to 2D image coordinates, which will allow us to define intrinsic and extrinsic parameters for the camera.

2.3.1 Extrinsic Parameters

N is a 4 by 4 displacement matrix accounting for camera position and orientation. It expresses the displacement from the world coordinate system ($X\ Y\ Z$ centered at O) to

the camera-centered coordinate system (X' Y' Z' centered at optical center C) by a translation followed by rotations around axes, shown in Figure 2.3.

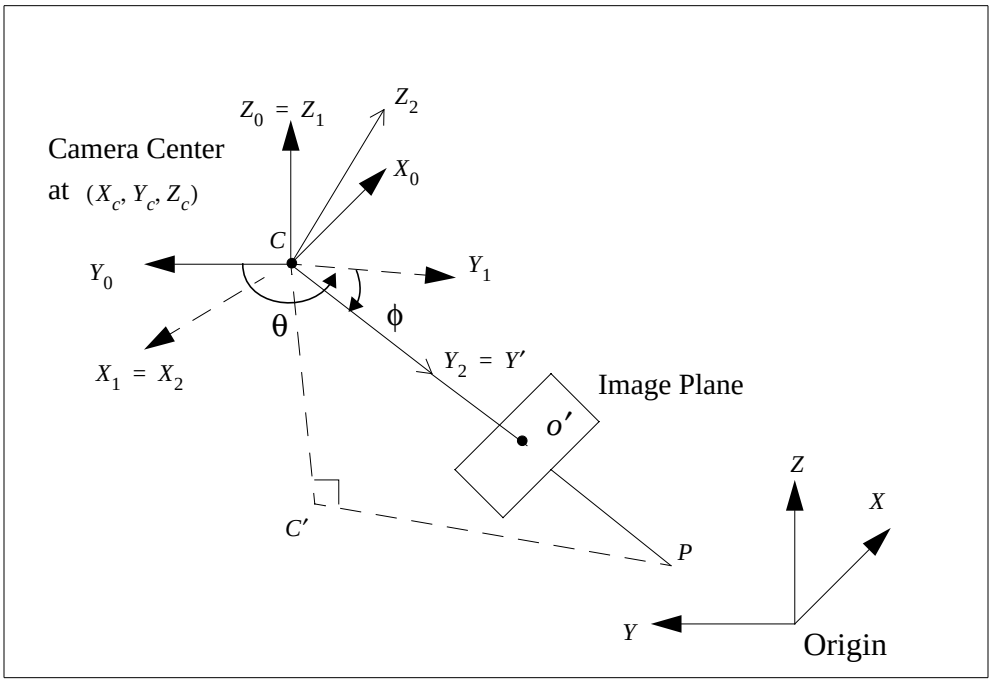

Figure 2.3 From World Coordinate System to
Camera-Centered Coordinate System

First, consider a camera center C located at (X_c, Y_c, Z_c) (measured in the X Y Z coordinate system) looking along a line of sight $CO'P$. P is the point at which the line of sight pierces the $X-Y$ plane and O' is the point at which the line of sight intersects the image plane (see Figure2.2). Let $C'P$ represent the projection of CP on the $X-Y$ plane. We can transform the x, y, z coordinates of a point to x', y', z' coordinates by choosing an

X', Y', Z' system (Camera Centered Coordinate System) of axes centered at C by the following two steps:

a) Move the origin

First we move the origin from O to C, leaving the axes' directions the same, we get the coordinate system $X_0\ Y_0\ Z_0$ (See Figure 2.3). This can be accomplished with a 4×4 displacement matrix D in homogeneous coordinate system.

$$D = \begin{bmatrix} 1 & 0 & 0 & -X_c \\ 0 & 1 & 0 & -Y_c \\ 0 & 0 & 1 & -Z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

b) Rotating axes

Pan: Let $C'P$ represent the projection of CP on the $X-Y$ plane, rotate the $X_0 - Y_0$ plane around the Z_0 axis by an angle θ such that the new Y axis (Y_1) is parallel to $C'P$ and is in the same direction as $C'P$. Now we have the coordinate system $X_1\ Y_1\ Z_1$ (See Figure 2.3). If we take X_0 to Y_0 seen from the positive Z_0 as the positive direction of rotation, the transformation matrix R_1 is:

$$R1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Tilt: Rotate $Y_1 - Z_1$ plane around the axis X_1 by an angle ϕ to align the new Y axis (Y_2) with CP. The positive direction of rotation is from Y_1 to Z_1, the transformation matrix R_2 that does this is:

$$R_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Swing: Now the axis Y_2 aligned along the line of sight and pointing towards the image plane at a distance F from the camera center C. The projection of the axis X_2 and axis Z_2 on the image plane is X_2', Z_2', shown in Figure 2.4.

As the axes of the image coordinate system are $U\ V$ and the original is I, now we rotate $X_2 - Z_2$ around the Y_2 axis by an angle ψ, we get the coordinate system $X'\ Y'\ Z'$.

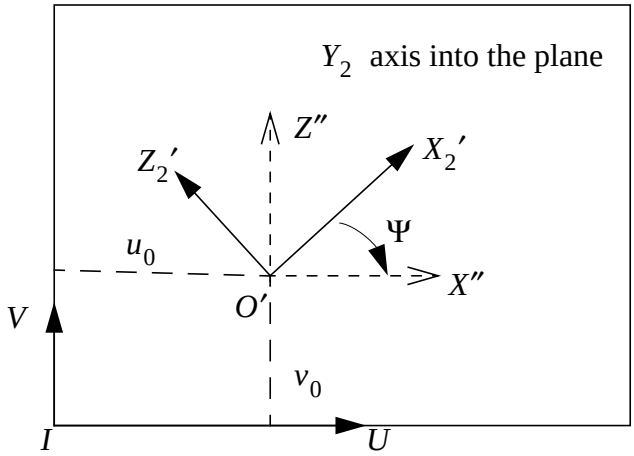

Figure 2.4 Image Coordinate Systems

ψ is the angle between the X_2' axis and the X'' axis - the projection of the X' axis on the image plane, X'' is aligned parallel to the U axis. The rotation will leave the projection of the Z' axis on the image plane - Z'' either aligned in the same direction as the V axis or parallel to it but in the opposite direction. The matrix R_3 that accomplishes this is:

$$R_3 = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

The resultant coordinate system $X'\ Y'\ Z'$ is the camera centered coordinate system. The image plane itself is at $y' = F$. The coordinates x', y', z' (in camera-centered coordinates) of a point x, y, z (in world coordinates) is given by

$$w \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = [N] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.8}$$

in this case $w = 1$, and $[N]$ is a 4 by 4 matrix equals to:

$$[N] = [R_3]\ [R_2]\ [R_1]\ [D] = [R]\ [D] \tag{2.9}$$

$X_c, Y_c, Z_c, \theta, \phi, \psi$ are called six extrinsic parameters, three for translation and three for rotation, which define the transformation from the world coordinate system to the camera-centered coordinate system.

2.3.2 Intrinsic Parameters

The matrix N expresses what happens when we change the world coordinate system to camera-centered coordinate system. Now as importantly we consider the matrix M which expresses what happens when we change the origin of the image coordinate system and the units on the U and V axes. The situation is shown in Figure 2.4. M is a 3 by 4 matrix.

The old image coordinate system is centered at the intersection O' of the optical axis with the image plane, and it has the same units on both axes. We go from the old image coordinate system to the new image coordinates system, which has its origin at a point I in the image (usually one of the corners) and will sometimes have different units at both axes. If we denote the original of the old coordinates system O' by u_0 and v_0, the scaling factor from the old coordinate system to the new one is k_u, k_v on both axes. As from previous discussion we know that the Y' axis of the camera centered coordinate system is pointing the image plane, and the projection of the X', Z' axis on the image plane is X'', Z'', by the perspective transformation, an image point coordinates in the X'' Y'' coordinate system on the image plane are:

$$x'' = \frac{x'F}{y'} \quad \text{and} \quad z'' = \frac{z'F}{y'}$$

Now we convert the measurement units to some different units on both axes, scale

by k_u in the X'' axis and k_v in the Z'' axis. Finally, translate the origin to I. If the coordinates of O' are u_0 and v_0, in the $U-V$ system we get:

$$u = u_0 + \frac{k_u x' F}{y'}, \quad v = v_0 + \frac{k_v z' F}{y'}$$

Using the homogeneous coordinate system this can be written as:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [M] \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.10}$$

where

$$[M] = \begin{bmatrix} k_u F & u_0 & 0 & 0 \\ 0 & v_0 & k_v F & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} k_1 & u_0 & 0 & 0 \\ 0 & v_0 & k_2 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.11}$$

Let $k_1 = k_u F$, and $k_2 = k_v F$, the parameter k_1, k_2, u_0 and v_0 are called camera intrinsic parameters, as they do not depend on the position and orientation of the camera in space. Knowledge of the intrinsic parameters allows us to perform metric measurements with a camera.

Now the transformation matrix T can be expressed by camera parameters: $[T] = [M]\,[N]$. M is expressed in terms of 4 parameters k_1, k_2, u_0 and v_0 called intrinsic parameters. N is expressed in terms of 6 parameters $X_c, Y_c, Z_c, \theta, \phi, \psi$ called

extrinsic parameters. Thus the whole image formation process can be expressed by these camera parameters only.

Chapter 3

Feature Detection

3.1 Introduction

Feature detection is one of the most important areas in computer vision. A great deal of effort has been spent by the computer vision community on this problem, and in particular on the problem of edge detection. Comparatively, there are fewer reports in the literature about corner detection. In vision, an edge is defined as a one-dimensional discontinuity of image brightness function, a corner is defined as a two-dimensional discontinuity of the image brightness function.

Why do we chose to use corners in our system? As we have no knowledge of 3D geometry of the objects, or use 3D reconstruction method to get such information, we want to use some features which not only abound in natural and man-made scenes, but also can convey significant information to us. Corners in image are attractive to work with

as they are likely to correspond to real 3D structure, such as corners of real objects and surface markings which usually abound both in natural and man-made scenes. Straight edge features more are suited to man-made environments than natural environments. Although curving edges are abundant in natural scenes, they can be temporally unstable. Corners in an image are easy to detect and simple to track through the sequence of images (while still in view) when the camera moves from one location to another and lighting conditions of the scene change somewhat; localization of corners in two dimensions can give good repeatability and can be done accurately, where an image decomposition into straight-line fragments can be highly erratic. The detection of corners is spatially and temporally a local operation, it is comparatively computational cheap; and most importantly as they usually are associated with the corners in real objects, there are easy to handle in 3D as a pragmatic consideration and they can be measured with little difficulty.

Corners as features are very important in computer vision. As these features can be used to identify objects in the scene, they are popular tokens for image analysis, for stereoscopic matching, displacement vector measuring, etc. An ideal corner in an image looks like Figure 3.1, it has an infinite sharpness and a clear boundary, while in practice corners are typically more rounded, blunted, blurred and ragged.

As corners in image are detected based on image intensity discontinuities, it can be caused by surface discontinuity in 3D space, texture changes on the objects or by some shadows cast by other objects etc., and therefore it may not associated with a corner in real scene, which is a junction of three or more surfaces. Similarly, a corner in 3D due to light

condition, view angles etc. may not turn up to be a corner in image. Here, in order to distinguish between corners in the image and corners in the real world, we refer the former as 2D corners, and the latter as 3D corners. Feature points is the more general term which is used here interchangeably with the term corners. In our system, we only interested in those 2D corners which have corresponding 3D corners in the scene, and try to establish relations between them. Thus an accurate localization of these features both in 2D and 3D is of great interest.

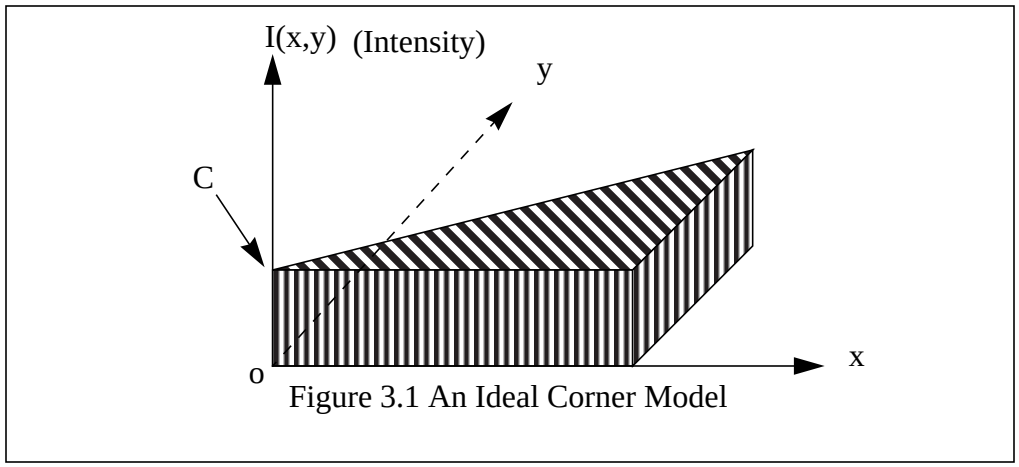

Figure 3.1 An Ideal Corner Model

3.2 Corner Detection Algorithms

The most popular image features used are those based on image intensity discontinuity. An edge, a 1D discontinuity, and a corner, a 2D discontinuity. In almost all the feature detection algorithms, the image intensity gradient at corners and edges is used, to a greater or lesser degree, as an intuitive measure of feature strength. In other words, the higher the local gradient, the greater the “cornerness” or “edgeness” of a feature, and so the higher

the confidence that a genuine corner or edge has been found. A simple thresholding technique is then often used to discard out outliers and non-salient features.

Several approaches to the problem of detecting corners have been reported in the literature in the last years. They can be broadly divided into three group:

The first category of method is to extract edges as a chain code (Figure 3.2), then search for points having maximal curvature [Asada 86]; variants are scanning the chain with a moving line segment which spans several links, the angular differences between successive segment positions are used as a smoothed measure of local curvature along the chain [Freeman 77]; or performing a polygonal approximation on the chain and then search for the line segment intersections [Horaud 90]. The computational cost is a great concern in these methods.

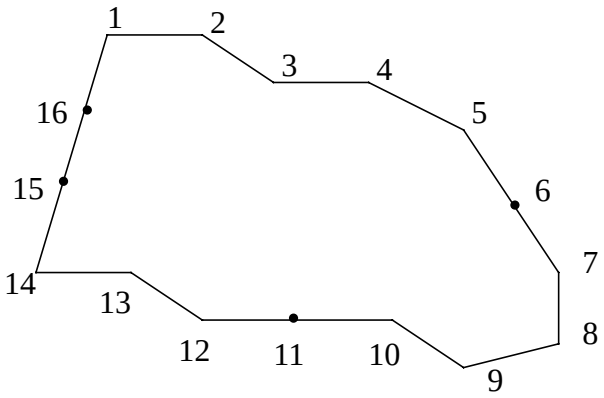

Figure 3.2 Searching Corners from a Chain

The second group of methods works directly on a grey-level image. These techniques are based either on heuristic techniques [Moravec 77], or apply a differential oper-

ator measuring gradients and curvatures of image intensity surface, then select points that are corners by a second operator that is often a thresholding scheme. Among the most popular corner detectors are those proposed by Kitchen [Kitchen 82], Wang [Wang 92] and Harris [Harris 88, 92].

As this approach seems better suited for our purpose we will select a corner detection method in this group, we will take a look at several of the best known corner detection methods in this group.

In [Moravec 77], Moravec developed the idea of using “points of interest”. They are defined as occurring when there are large intensity variations in every direction. This definition is realized by computing an unnormalized local autocorrelation in four directions and taking the lowest result as the measure of interest. This response is thresholded and then local non-maxima are suppressed. However, some problems with the Moravec operator have been identified (principally by the proponents of the Plessey detector, which builds on the same feature definition): the response is anisotropic, the response is noisy, and the operator is sensitive to strong edges.

In [Kitchen 82], Kitchen used a local quadratic surface fit to find corners. The parameters of the surface were used to find the gradient magnitude and the rate of change of gradient direction; the product of these quantities was used to define “cornerness”, and local maxima were reported as corners.

In [Wang 92], Wang developed the curvature based methods mentioned. As well as requiring that curvature be a maximum and above a threshold, they require that gradient perpendicular to the edge be a maximum and above threshold. Also false corner response suppression is performed to prevent corners being wrongly reported on strong edges. The corners are found at different smoothing levels allowing an estimation of the corner positions at zero smoothing. This is a very promising method, although the reliability of detection by this method is not clear.

In [Harris 88, 92], Harris described which has become known as the Plessey feature point detector. It is built on similar ideas to the Moravec interest operator, but the measurement of local autocorrelation is estimated from first order image derivatives. The variation of the autocorrelation over different orientations is found by calculating functions related to the principle curvatures of the local autocorrelation. This well conditioned algorithm gives robust detection. We will discuss this algorithm in detail later.

Algorithms in the second groups are global and quite efficient approaches, however it has been shown that the best known algorithms (Harris method) of this family yield a precision only a few pixels in the positioning.

A third group of approach is emerging, mainly characterized by using model-based corner detection, they match a part of the image containing a corner against a predefined corner model (See Figure 3.3). Once the fitting is accomplished, the position of the corner

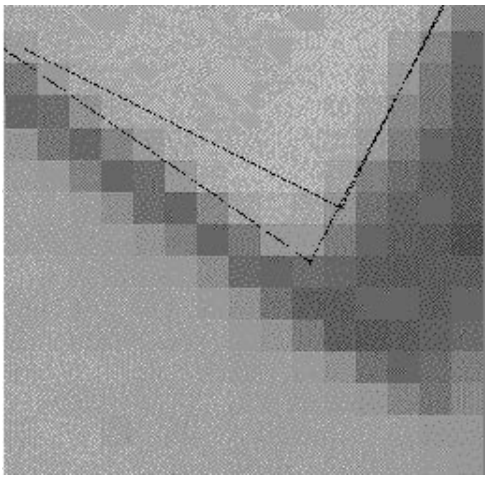

Figure 3.3 Fitting a Corner with a Model (first approximation and final estimation)

in the image can be deduced by the knowledge of the corner position in the image [Deriche 90, 91, 93], [Rohr 92], [Blaszka 94].

In [Derich 90, 93], Deriche considered a corner model and studied analytically its behavior once it has been smoothed using the well-known Gaussian filter. In these paper, the author clarifies completely the behavior of some well known cornerness measure based approaches used to detect these points of interest. In particular, most of the classical approaches presented in the second group have been shown to detect points that do not correspond to the exact position of the corner. A new scale-space based approach has been proposed in order to correct and detect exactly the corner position.

In [Rohr 92], Rohr proposed an general analytical model and fitted it directly to the image intensities to determine the position of grey value structures to sub-pixel accuracy and also additional attributes such as the width of the grey value transitions.

But the main drawback of the third group is the high computational cost, and the difficulty to perform the methods in a purely automatic manner.

We can summarize now some of the qualities expected from a good corner detector:

Good detection. There should be a minimum number of false negatives and false positives.

Good localization. The feature location must be reported as close as possible to the correct position.

Response. Only one response to a single feature.

Speed. The algorithm should be fast enough to be usable in the final image processing system.

Consistency. Most importantly, if corners are to be used as features, upon which subsequent processing is to be based, they must be detected consistently, that is have good repeatability, even for natural scenes.

There are some trade-off among these criteria. Having compared several algorithms just mentioned before, we choose the Harris detector in our system, because it has the best repeatability, means most points can be reliably detected at different scale, rotational change, and brightness change. This is very important for image matching, as the detection quality influences the matching quality.

3.3 Harris Corner Detection in Detail

The Harris corner detector was developed as a set of enhancements to the Moravec inter-

est operator. The problem of detecting corners can be analyzed in terms of the curvature properties of the local image brightness autocorrelation function where the curvature information can be represented by the Hessian matrix. The autocorrelation is useful for characterizing how the brightness values change in the neighborhood of a location. At a corner or an isolated brightness peak all shifts will result in a large change in the autocorrelation function at that location. A version of the brightness spatial change function for a small shift(x,y) can be written:

$$E(x, y) = \sum_{u, v} w_{u, v} \left| I_{x+u, y+v} - I_{u, v} \right|^2 \tag{3.3.1}$$

Where E is the average changes of image intensity produced by a shift (x, y), w specifies the image window, I denotes the image intensity.

A first order approximation of E is given by:

$$E(x, y) = Ax^2 + By^2 + 2Cxy \tag{3.3.2}$$

where A, B, C are approximations of the second order directional derivatives, which are computed by X^2, Y^2, XY convolving with w respectively:

$$A = X^2 \otimes w, \quad B = Y^2 \otimes w, \quad C = X\,Y \otimes w$$

w is a smoothing circular window, in this case a Gaussian:

$$w = \frac{1}{2\pi\delta^2} e^{-\frac{x^2 + y^2}{2\delta^2}}$$

X and Y are the first order directional derivatives, which are approximated by convolving the intensity image I with a finite difference operator of the form $[1, 0, -1]$ in X direction and $[1, 0, -1]^T$ in Y direction:

$$X = I \otimes [1, 0, -1] \approx \frac{\partial I}{\partial x}, \qquad Y = I \otimes [1, 0, -1]^T \approx \frac{\partial I}{\partial y}$$

Then (3.3.2) can be rewritten as:

$$E(x, y) = [x, y]\, M\, [x, y]^T \tag{3.3.3}$$

where the matrix M is an approximation of the Hessian matrix for E:

$$M|_{(x,y)} = \begin{bmatrix} A|_{(x,y)} & C|_{(x,y)} \\ C|_{(x,y))} & B|_{(x,y)} \end{bmatrix}$$

E is closely related to the image's local autocorrelation function. The principal curvatures of the image brightness autocorrelation at a point can be approximated by the eigenvalues of the approximated 2×2 Hessian matrix M defined at that point. If both eigenvalues of this matrix are large, so that the local correlation function is sharply peaked, then a shift in any direction will increase E, that is the local grey patch cannot be moved in any direction on the image without a significant change in grey level. This indicates the window is at a corner. The response value is computed, and then any values that are not local maxima are suppressed. This leads to a sparse corner point map for each image. The determinant of the approximated Hessian matrix, $det[M]$, is proportional to the product of the principal curvatures. The Harris-Steven's corner detector is given by the

following operator where a large value of R signals the presence of corner (Equation 3.3.4). Then values which are not local maxima are suppressed, this leads to a sparse corner point for each image. Finally we use a threshold to get only some strongest corners for each image.

$$R(x, y) = det[M|_{(x,y)}] - k\, trace^2[M|_{(x,y)}] \tag{3.3.4}$$

where, $det[M] = A + B$,

$$trace[M] = AB - C^2$$

A positive value for scalar k can be used to ensure that the Gaussian curvature approximate $det[M]$ is valid, Zhang *et al.* [Zhang 95] specify $k = 0.04$ which is the best value for our cases as well.

3.4 Sub-Pixel Accuracy

In Figure 3.4, suppose image grey values are only available at pixel level, after computing

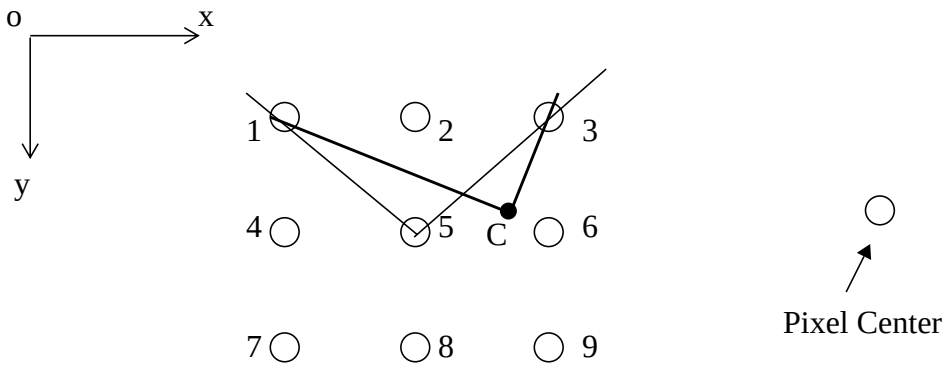

Figure 3.4 a Corner with Subpixel Accuracy

the directional intensity gradient for every pixel, we obtain a local maxima at pixel 5. In fact the real corner position may be between pixels. If the grey values between pixels were available, after computing the directional greatest intensity gradient in this area again, we may get a local maxima at position C, which closer to the true position of the corner. This is what we mean by sub-pixel accuracy.

This is important to consider because this will happen in real situations. For example, in our system, when the camera moves, the movement for some corners in the image sequences may less than one pixel. If we have a corner detector which can only give pixel accuracy, then we can not get the accurate position for these corners. So we need a corner detection which can give sub-pixel accuracy.

As Harris algorithm only allow us to get a corner position up to a pixel precision, in order to recover the corner position to a sub-pixel position, some extra effort should be made. In order to concentrate on the area where they might be corners, we first use the Harris corner detector at the pixel level, then we interpolate image grey values in the areas near detected corners (See Figure 3.5), in this case among 3 by 3 pixels.

For the interpolation, we know that the ideal 1D interpolation filter in frequency space is the box, but the inverse transform of this finite-support spectrum is an infinite impulse response *sinc* function. We cannot convolve a signal with this *sinc* function in a practical system as it requires us to have access to the signal from negative to positive infinity. What we prefer is a filter that comes close to the box in frequency space, but still has a finite, reasonable width in signal space. Because of the inverse relationship between

the width of a signal and its Fournier transform - the narrower one becomes, the wider the other spreads, we cannot really hope to find a very boxlike filter with a small and finite impulse response. One practical method is to design a filter that drops off to a very small value outside of some interval in both spaces. A popular choice is the Gaussian bump, as it drops off “almost to zero” at some distance from the center in both spaces, therefore it approximates a signal with finite support in both spaces. The same principle applies to 2D interpolation, in our system, we choose to use a 2D Gaussian interpolation filter to reconstruct the grey values in the interested areas.

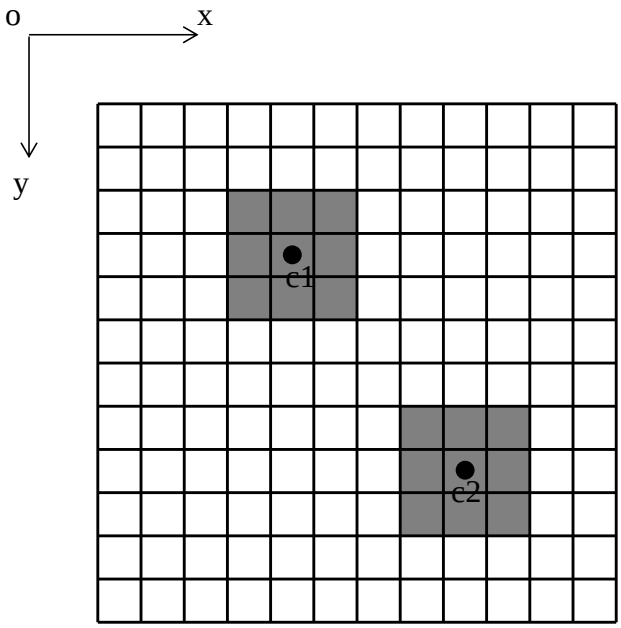

Figure 3.5 Grey Value Interpolation around the Interest Points

The 2D Gaussian interpolation filter is:

$g(x,y) = \frac{1}{k_1} e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\delta^2}}$ where (x_0, y_0) is the center of the filter, as

Gaussian filter has the property of *separability*, it can be written as $g(x, y) = g_1(x)\ g_2(y)$ where $g_1(x) = \frac{1}{\sqrt{k1}} e^{-\frac{(x-x_0)^2}{2\delta^2}}$ and $g_2 = \frac{1}{\sqrt{k1}} e^{-\frac{(y-y_0)^2}{2\delta^2}}$. This leads to ease of implementation. a 2D implementation filter can be performed by a 1D interpolation filter followed by another 1D interpolation filter acting perpendicularly to the first one.

After the interpolation the corner detection is used again at these interpolated areas, the corner position up to the accuracy of 1/10th of a pixel can be detected.

Chapter 4

Feature Correspondences

4.1 Introduction

In our system, feature correspondences between 2D points and 3D points in the world space need to be established, as well as feature correspondences between 2D points in different images. These are done by two different steps, which will be described in detail in this chapter. Once the matching hypotheses are generated, the camera parameters can be computed based on these correspondences. Since even a single mismatch can have a large effect on the result, special care has been taken in the matching process.

For 2D-3D feature correspondences, first a corner detector is used to get some feature points in the images, for the first frame. Then 3D world coordinates are input by hand for some 2D points if they correspond to some real corners in the world, then a 2D-3D fea-

ture correspondences list is established, for the following frames, by matching the image with the previous one, the 2D-3D feature correspondences list can be maintained automatically. Later on the perspective transformation matrix can be determined from the 2D-3D feature correspondences list.

4.2 Related Work

In this chapter we put our focus on 2D-2D feature correspondences. This kind of matching is the same as image matching, except that it only matches some interesting features, not all the pixels in the image. Any method used for image matching can be used for matching. So we should analyze the best known image matching methods therefore to find one that is most suitable for our system.

The image matching method which has been used for the longest is signal correlation, Faugeras [Faugeras 92] compares the different methods. To obtain satisfactory results the signal has to be taken under very similar condition. If we have to deal with non-trivial image rotation this method fails.

A lot of image matching or feature matching method try to use the geometric constraints, like epipolar constraint, in reducing the false matches. Zhang [Zhang 95] proposes a robust approach for image matching by exploiting the only available geometric constraint, the epipolar constraint. The images are uncalibrated, the motion between them is unknown, also the camera parameters are known too. The idea is to use classical tech-

niques and relaxation methods to find an initial set of matches, and then use a robust technique-the Least Median of Squares (LMedS) to discard false matches. Thus the epipolar geometry can be estimated accurately using a well adapted criterion from these matches. In turn by using the recovered epipolar geometry, more matches are eventually found.

Schmid [Schmid 95], [Schmid 96] presents a matching method which is sufficiently general in that the images can be taken under very different conditions. This method is based on invariants of the luminance function. It uses the differential invariant scheme to locally characterize the brightness function over a set of point feature. Having calculated the derivatives of a function on a point up to Nth order (N=3 has been used), differential invariants can be calculated. The set of invariants used is stacked in a vector denoted by Vi, it is invariant to image rotation and translation, at the same time a multi-scale invariant representation affords scale invariance. After points of interest are characterized by Vi this vector simple brute force matching compare the Malinois distance between the vectors of invariants to get the final result.

It is well known however, that the calculation of the derivatives is ill-conditioned, so it is necessary to calculate the derivatives on smoothed data to reduce noise. Smoothing the signal improves the stability but it may damage useful information. In case of illumination change, some non linear effect can occur, and this method may fail in case of partial occlusion.

In our system, the image sequences usually are taken under very similar conditions, the camera movement or rotation between two consecutive frames is small, plus usually there are less than 20 feature points in each frame whose 3D coordinates are available, so we can choose the classical correlation method for feature correspondence, as it is fast and efficient. Although other methods we have discussed before might give better matching result, there are relatively complicated and less efficient.

4.3 2D - 2D Feature Correspondences

The use of correlation as a measure of similarity between two signals is a well known technique and it is commonly used in stereo vision to solve the correspondence problem. Template windows are chosen from the first image as 2D information samples to be matched in a region of interest of the second image.

In [Faugeras 93], Faugeras gave four criteria for correlation after he had extensively tested the four criteria, and concluded that the method which uses the sum of normalized mean-squared differences of grey level values performs best in practice, because “it is the most invariant to affine transformations of the images which may result from slightly different settings of the cameras, and it gives good experimental performance to varying lighting conditions”.

In this matching stage, our inputs are two images containing feature points, the aim of the matching is to find which corners of each image are the projections of the same cor-

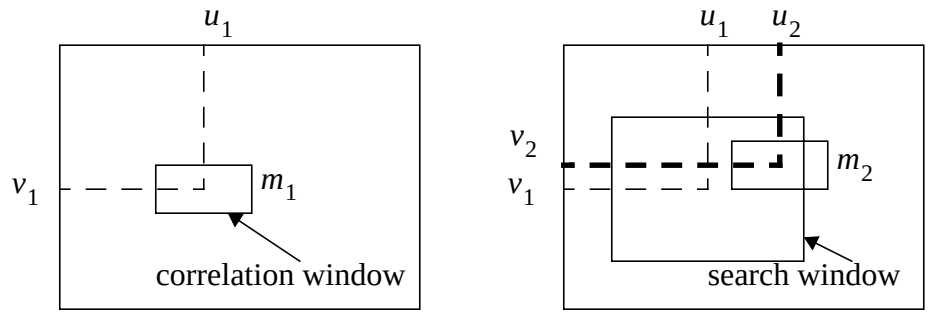

Figure 4.1 Correlation

ners of the 3D object. The output is a correspondence list between the feature points of the two images.

As shown in Figure 4.1, given a high curvature point m_1 in image 1, we use a correlation window of size $(2n+1) \times (2m+1)$ centered at this point. We then select a rectangular search area of size $(2d_u + 1) \times (2(d_v + 1))$ around this point in the second image, and perform a correlation operation on a given window between point m_1 in the first image and all high curvature points m_2 lying within the search area in the second image. The search window reflects some a priori knowledge about the disparities between the matched points. This is equivalent to reducing the search area for a corresponding point from the whole image to a given window. The correlation score is define as:

$$c = \frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m}\left(\left(I_1(u_1+i, v_1+j) - \overline{I_1(u_1, v_1)}\right) - \left(I_2(u_2+i, v_2+j) - \overline{I_2(u_2, v_2)}\right)\right)^2}{\sqrt{\sum_{i=-n}^{n}\sum_{j=-m}^{m}\left(I_1(u_1+i, v_1+j) - \overline{I_1(u_1, v_1)}\right)^2 \sum_{i=-n}^{n}\sum_{j=-m}^{m}\left(I_2(u_2+i, v_2+j) - \overline{I_2(u_2, v_2)}\right)^2}}$$

$s = max(0.1 - c)$

where $\overline{I_k(u,v)} = \frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m} I(u+i, v+j)}{(2n+1)(2m+1)}$ is the average at point (u,v) of I_k $(k = 1, 2)$.

We refer to this score s as a correlation score, where I_1 and I_2 are the left and right image intensities. $\overline{I_1}$, $\overline{I_2}$ are their average value over the correlation window and dx, dy represent the pixel displacement in the template window.

In our implementation, $n = m = 7$ for the correlation window, as it gives the best matching result. For the search window, du and dv are set to a quarter of the image width and height, respectively, it is large enough (half of the whole image) to trace feature points between image frames.

4.4 Validity Check

As mentioned in [Faugeras 93], the probability of a mismatch goes down as the size of the correlation window and the amount of texture increase. However, using large windows leads to the performance problem. Here a definition of valid measure in which the two images play a symmetric role is used, which allows us to reduce the possibility of error even when using very small windows. The correlation is performed twice by reversing the roles of the two images and we consider as valid only those matches for which the

same optimal correlation is measured at corresponding points when matching from I_1 into I_2 and I_2 into I_1.

For example, given a point P_1 in I_1, let P_2 be the point of I_2 corresponding to P_1 such that the windows centered on P_1 and P_2 yield the optimal correlation measure. The match is valid if and only if P_1 is also the point that maximizes the scores when correlating the window centered on P_2 with windows of I_1 that centered on P_1.

In our system, a major concern of our work is the accuracy of the correspondences. Higher accuracy of the correspondences leads to higher accuracy of the camera parameters. Since detecting the same corners arising from the same scene corner from images taken under different illumination or from different viewpoints is known to be unreliable, no matching method can achieve 100% correct rate, and some incorrect matches may be accepted by the matching algorithm. In this case the wrong 3D coordinates will be assigned automatically to the feature points in the next frame and those false matches might severely affect the value of the transformation matrix (we will discuss this in Chapter 5). Of course the camera parameters based on decomposing the transformation matrix will be quite different as well (these can be seen at the test results in Table 6.3). One solution is to eliminate the false matches, which we currently do manually.

Chapter 5

Recovering Camera Parameters

The problem of the determination of the camera location and orientation as well as camera intrinsic parameters from 2D-3D correspondences is a very basic problem in image analysis, and it is extremely important for practical applications. Most of the earlier or recent studies in the field assume that calibration of the camera is known, which means that the camera intrinsic parameters are known, Haralick *et al.* [Haralick 89] cite a German dissertation from 1958 that surveys nearly 80 different solutions to pose estimation (camera extrinsic parameters). Moreover, the analysis of pose estimation in the case of an uncalibrated image sequence has already been developed by several authors, considering point and/or line correspondences. These studies are motivated by the fact that we must not consider the camera is calibrated in an active vision, and it is not possible to self-calibrate the

camera when zooming or modifying during the video capture the intrinsic calibration parameters.

5.1 Related Work

From Chapter 2, we know that the determination of camera location and orientation means to relate the camera-centered coordinate system to the world coordinate system by rotations fallowed by a translation. Liu *et al.* [Liu 88] represent the relationship between two points in these different coordinate system by: $p = Rp' + T$, where R is a 3 by 3 rotation matrix and T is a 3 element vector, then compute the rotation matrix and the translation vector separately using line correspondences. As for point correspondences, they first derive lines from points, then use the same line correspondences method to get R and T. The authors provided both linear and non-linear algorithms. But in their method, the authors assumed that the focal length is known and the optical axis of the camera passes through the center of the image, which are not often the case. Also the authors noticed that their method worked well only when the three rotation angles are less than 30^0.

Vieville *et al.* [Vieville 93] considered the motion of a rigid object or the ego-motion of the camera observing a stationary scene determined by points and/or lines correspondences. They not only assumed that the camera is not calibrated, but furthermore lifted the restriction that the intrinsic parameters of the camera are constant for most uncalibrated cases, and they used the standard pinhole model for a camera. In their paper, the authors

first gave some precise definition for tokens and motion of points and lines through rigid displacements, then from an algebraic point of view using a special representation, called Qs-representation, the authors related these parameters to points and/or lines 2D correspondences measured in an image sequence for a system of cameras without calibration. Then the authors proposed two different algorithms to estimate these parameters. Implementing such algebraic representation is not easy. The authors define a criterion to estimate the Qs-representation, but "this is a rather huge criterion". In order to dramatically simplify the amount of calculations and to accelerate the convergence of the estimation process, the problem has to be decomposed into two sub-optimal problems and a non-trivial architecture has to be built in order to gain an efficient algorithm. The results are not promising in general cases, and some additional hypotheses have to be used, such as knowing the intrinsic parameters for the first frame. It is also sensitive to small errors, and since the radius of convergence of the algorithm is very small, extra steps are need to be taken to ensure the convergence of the method. In addition that the algebraic equations are really complicated and the authors omitted a lot of intermediate calculations, making the paper hard to follow.

Ganapathy [Ganapathy 84] showed how the location and orientation of the camera, as well as the other parameters of the image formation process can easily be computed from the homogeneous coordinate transformation matrix. He first used a noniterative method for solving this kind of problem. We will discuss this method in detail later. In [Strat 84] the author tried to compute the same parameters but from another point of view.

He claimed that Ganapathy's method is an algebraic one that provides little insight into the underlying geometry, and tried to use a geometric method to solve for the same parameters. The author admitted nevertheless that "While we do not expect this technique to be any more robust than that of Ganapathy, we do feel that its geometric interpretation provides useful clues as to when it will be dependable". Like Ganapathy, this method is also a non-iterative one, with a small computational burden.

In [Huang 94], the author gave a review of motion from feature (points or lines) correspondences, including 2D-2D, 2D-3D and 3D-3D feature correspondences. With 2D-2D point correspondences, the author summarized four very similar methods, including Ganapathy's method we will use in this project, but he did not evaluate them, or express opinion about which one is better. Although some methods compute the camera location directly from a set of landmarks with known 3D locations, they need to know the focal length and piercing point, which is not the case with Ganapathy's method.

In our system, the Ganapathy method was chosen to recover the camera parameters when the transformation matrix is known. Our method is based on decomposing the transformation matrix. A least squares method is used to estimate the transformation matrix (Equation 2.3) which project 3D points to 2D image points, then Ganapathy's method is used to retrieve the camera parameters from this matrix.

5.2 Getting the Camera Transformation Matrix

We assume that we have a set of points whose coordinates in three dimensions $X\ Y\ Z$ are known, and whose locations in two dimensions $U\ V$ are also known. In addition to these points, we also know that the view is a perspective projection, and therefore the transformation of the three-dimensional coordinates of a point to the two-dimensional coordinates can be expressed compactly as a 3 by 4 transformation matrix using homogeneous coordinate:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = w' \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2.1}$$

Notice that there are 12 elements in the transformation matrix, and that the measurements from the images are not the values x', y' and w' shown in (5.2.1), but the ratio $U = x'/w'$ and $V = y'/w'$. In fact, we have no way of knowing the value of w', and different input points $X\ Y\ Z$ may give different values of w'.

Now we write out the separate equations implied in the matrix expression in (5.2.1):

$$\begin{aligned} T_{11}X + T_{12}Y + T_{13}Z + T_{14} &= w'U \\ T_{21}X + T_{22}Y + T_{23}Z + T_{24} &= w'V \\ T_{31}X + T_{32}Y + T_{33}Z + T_{34} &= w' \end{aligned} \tag{5.2.2}$$

By substituting the value of w' found in the third equation into the first two equations and grouping terms, we get:

$$(T_{11} - T_{31}U)\,X + (T_{12} - T_{32}U)\,Y + (T_{13} - T_{33}U)\,Z + (T_{14} - T_{34}U) = 0 \qquad (5.2.3)$$

$$(T_{21} - T_{31}U)\,X + (T_{22} - T_{32}V)\,Y + (T_{23} - T_{33}V)\,Z + (T_{24} - T_{34}V) = 0$$

If we know the location of several points in (X_i, Y_i, Z_i) and know their positions in the view (U_i, V_i), then (5.2.3) gives two equations per point for the twelve unknowns T_{ij}. For 6 points, the minimal number necessary to compute all the T_{ij}, subscripts a through f, the matrix looks like:

$$\begin{bmatrix}
X_a & Y_a & Z_a & 1 & 0 & 0 & 0 & 0 & -U_aX_a & -U_aY_a & -U_aZ_a & -U_a \\
X_b & Y_b & Z_b & 1 & 0 & 0 & 0 & 0 & -U_bX_b & -U_bY_b & -U_bZ_b & -U_b \\
X_c & Y_c & Z_c & 1 & 0 & 0 & 0 & 0 & -U_cX_c & -U_cY_c & -U_cZ_c & -U_c \\
X_d & Y_d & Z_d & 1 & 0 & 0 & 0 & 0 & -U_dX_d & -U_dY_d & -U_dZ_d & -U_d \\
X_e & Y_e & Z_e & 1 & 0 & 0 & 0 & 0 & -U_eX_e & -U_eY_e & -U_eZ_e & -U_e \\
X_f & Y_f & Z_f & 1 & 0 & 0 & 0 & 0 & -U_fX_f & -U_fY_f & -U_fZ_f & -U_f \\
0 & 0 & 0 & 0 & X_a & Y_a & Z_a & 1 & -V_aX_a & -V_aY_a & -V_aZ_a & -V_a \\
0 & 0 & 0 & 0 & X_b & Y_b & Z_b & 1 & -V_bX_b & -V_bY_b & -V_bZ_b & -V_b \\
0 & 0 & 0 & 0 & X_c & Y_c & Z_c & 1 & -V_cX_c & -V_cY_c & -V_cZ_c & -V_c \\
0 & 0 & 0 & 0 & X_d & Y_d & Z_d & 1 & -V_dX_d & -V_dY_d & -V_dZ_d & -V_d \\
0 & 0 & 0 & 0 & X_e & Y_e & Z_e & 1 & -V_eX_e & -V_eY_e & -V_eZ_e & -V_e \\
0 & 0 & 0 & 0 & X_f & Y_f & Z_f & 1 & -V_fX_f & -V_fY_f & -V_fZ_f & -V_f
\end{bmatrix} \times \begin{bmatrix} T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{21} \\ T_{22} \\ T_{23} \\ T_{24} \\ T_{31} \\ T_{32} \\ T_{33} \\ T_{34} \end{bmatrix} = 0 \qquad (5.2.4)$$

Because the equations are homogeneous, the solutions will contain an arbitrary scale factor. In Ganapathy's method to compute the camera parameters, he assumed that T_{34} of the transformation matrix is unity. We will also choose T_{34} to be unity, move the last col-

umn to the other side of the equation, and solve for the remaining 11 T_{ij}.

In most cases, we have more than 6 points, and the system is overdetermined. A least-squares data-fitting technique is then used to obtain a solution. The formulation for the linear least squares problem can be written as:

Given $A \in \Re^{m \times n}$, $m \geq n$, $b \in \Re^{m}$ and equations $Ax = b$,

find $x \in \Re^{n}$ such that $\|b - Ax\|^2$ is minimized. (5.2.5)

The 2-norm corresponds to Euclidean distance, so there is a simple geometric interpretation of (5.2.5). Find the closest point Ax in range(A) to b, so that the norm of the residual $r = b - Ax$ is minimized as illustrated in Figure 5.1. Range(A) is the set of vectors that can be expressed as Ax for some x.

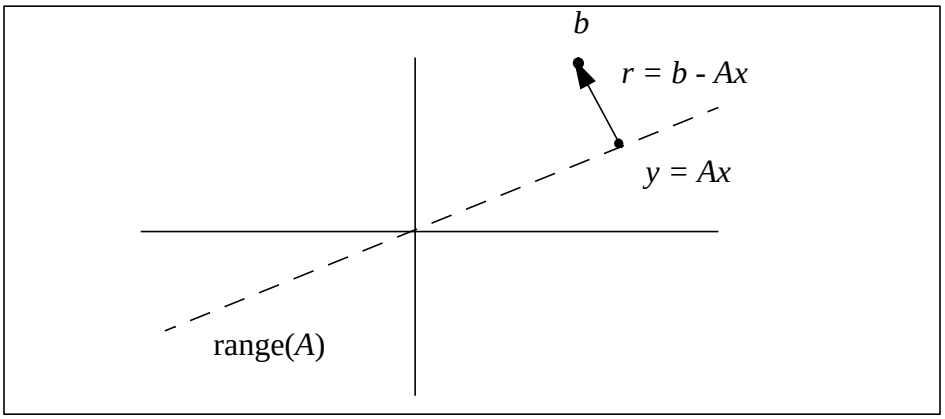

Figure 5.1 Formation of the least square problem

There are several methods to solve the least squares problem. We chose to use QR factorization method, described in [Strang 86] as “very fast” and “extremely stable”. So

we need at least 11 equations or 5 1/2 pairs of point correspondences for it to work, but most of the time, we will have an over-determined system of equations.

While using the least squares data fitting technique, we should put more attention to the selection of the matching point pairs. After the corner detection step, and the correlation matching described in chapter 4 and 5, we may find two types of outliers in the dataset:

1) Bad localization. There are localization errors of 2D corners in the image. They are assumed to exhibit Gaussian behavior, which is reasonable in practice. Generally the errors in localization for most 2D corners are small, but a few corners are possibly localized more than three pixels away from their real location. These corners will bring a lot of inaccuracy to the dataset and might severely degrade the accuracy of the estimation.

2) False matches. In the establishment of the correspondences, as we do not use any other constraints, for example geometric constraints, some matches are possibly false. These will spoil the estimation of the transformation matrix. The final estimation of the camera parameters may be very different from the real values as illustrated by the test results in Chapter 6.

Therefore, in order to get better results, some good matching pairs should be chosen to avoid the two kinds of outliers as we have just mentioned to get a more accurate transformation matrix for later use.

5.3 Recovering the Camera Parameters from the Transformation Matrix

The camera transformation matrix has an important role in computer graphics, and other fields as in stereo reconstruction, robot vision, photogrammetry, unmanned-vehicle guidance and image understanding. When the location and orientation of the camera are known, the camera transformation matrix that models the image formation process can easily be derived analytically. Here we are concerned with the inverse problem - given the transformation matrix, determine the camera location, orientation, scaling and translation parameters in the image plane from it. This problem is equivalent to the following problem: given an image that may have been enlarged both horizontally and vertically and clipped arbitrarily, determine the position and orientation of the camera used to take the picture from knowledge of the correspondences between known objects and their projections in the image. Any formulation of this problem results in a set of nonlinear simultaneous equations in several independent variables and hence the methods that have been proposed are iterative in nature and use hill-climbing or other well-known similar numerical techniques for the solution.

In the work we have already mentioned by Ganapathy and Strat, new and simple non-iterative analytical techniques are proposed that solve the camera location determination problem.

5.3.1 Decomposition of the transformation matrix

The problem addressed here involves decomposing the transformation matrix to arrive at the camera intrinsic parameters k_1, k_2, u_0, v_0 (see equation 2.11 in chapter 2) and the camera extrinsic parameters X_c, Y_c, Z_c θ, ϕ, ψ (see equation 2.4-2.2.7 in chapter 2).

These ten parameters are needed in order to determine the projection of an object onto the image: six parameters define the motion between the object and the camera, i.e. three angles of rotation around each axis (θ, ϕ, ψ) and three scalars for a translation (X_c, Y_c, Z_c), two parameters give the scale factor between the camera frame and the image frame (k_1, k_2), the last two parameters define the translation between the origin of the image origin and the intersection between the image plane and the optical axis (u_0, v_0).

From equation (2.9), if we reexpress R as:

$$R = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

then, we can obtain that:

$$[EXT] = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & h & i & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and we will get the following relations:

$$a = \cos\psi\cos\theta - \sin\theta\sin\phi\sin\psi, \qquad (5.3.1)$$

$$b = \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi, \tag{5.3.2}$$

$$c = -\cos\phi\sin\psi, \tag{5.3.3}$$

$$d = -\sin\theta\cos\phi, \tag{5.3.4}$$

$$e = \cos\theta\sin\phi, \tag{5.3.5}$$

$$f = \sin\phi, \tag{5.3.6}$$

$$g = \cos\theta\sin\psi + \sin\theta\sin\phi\cos\psi, \tag{5.3.7}$$

$$h = \sin\theta\sin\psi - \cos\theta\sin\phi\cos\psi, \tag{5.3.8}$$

$$i = \cos\phi\cos\psi, \tag{5.3.9}$$

and p, q, r are expressed in terms of $a, b, c, d, e, f, g, h, i$ as

$$p = -aX_c - cY_c - cZ_c, \tag{5.3.10a}$$

$$q = -dX_c - eY_c - fZ_c, \tag{5.3.10b}$$

$$r = -gX_c - hY_c - iZ_c \tag{5.3.10c}$$

As we know $T = [INT]\ [EXT]$, we can get:

$$T_{11} = (k_1 a + u_0 d)/q, \quad T_{12} = (k_1 b + u_0 e)/q,$$

$$T_{11} = (k_1 c + u_0 f)/q, \quad T_{12} = (k_1 p + u_0 q)/q,$$

$$T_{11} = (k_2 g + v_0 d)/q, \quad T_{12} = (k_2 h + v_0 e)/q,$$

$$T_{11} = (k_2 i + v_0 f)/q, \quad T_{12} = (k_2 r + v_0 q)/q,$$

$$T_{31} = d/q, \quad T_{32} = e/q, \quad T_{33} = f/q \tag{5.3.11 - 5.3.21}$$

Since we set T_{34} to 1, there are 11 unknown T_{ij}, we have eleven equations (5.3.11 - 5.3.21) in terms of 16 unknown $k_1, k_2, u_0, v_0, X_c, Y_c, Z_c, a, b, c, d, e, f, g, h$ and i. But 9 unknown $a, \ldots, i$ can be specified in terms of the 3 unknowns θ, ϕ, ψ (5.3.1 - 5.3.9), therefore we really have 11 equations and in only 10 unknowns, we can solve the equations and get the solution for the 10 unknowns. However, expressing $a, \ldots, i$ in terms of θ, ϕ, ψ is not only messy, but it makes the decomposition algorithm harder to understand. Hence, we will still use 16 unknowns but impose additional constraints. R is a pure rotation matrix, so it must be a "proper orthonormal matrix", it follows that $R^{-1} = R^T$ and R^{-1} is given by:

$$R = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \qquad R^{-1} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \tag{5.3.22}$$

this property along with the property that the determinant of R must be unity implies the following constraints:

$$a^2 + b^2 + c^2 = d^2 + e^2 + f^2 = g^2 + h^2 + i^2 = 1 \tag{5.3.23 - 5.3.25}$$

$$i = ae - bd, \quad g = bf - ce, \quad h = cd - af \tag{5.3.26 - 5.3.28}$$

They are sufficient to derive the other constraints we need, namely:

$$ad + be + cf = 0, \quad gd + he + if = 0 \tag{5.3.29 - 5.3.30}$$

We will use these constraints together with the other equations (5.3.1 - 5.3. 21) to decompose the matrix, and get the unknowns. In Ganapathy's paper, the author gave an ideal case of decomposition assuming that there are no errors in the T_{ij} terms. But in actual cases all terms T_{ij} have errors. In this case Ganapathy also gave a practical solution. The method relaxes the requirement that the terms a through i form an orthonormal matrix by dropping some of the (5.3.23 - 5.3.30) constraints. This method doesn't use a least squares fit or some such techniques, basically it derives relations from equations, and can compute each value directly from the equations. Ganapathy's paper gives the whole process of derivation and an example of computation.

Chapter 6

Experimental Results

This chapter presents the results obtained for a series of experiments for the task of retrieving camera parameters. Both synthetic and real data have been used to test the system, and the results show that it yields an accurate estimation in most cases.

6.1 Images taken with synthetic camera

The synthetic images were produced using “optik”- an experimental interactive ray tracing program developed by the University of Toronto’s Dynamic Graphics Project and the University of British Columbia’s Imager Lab [Amanatides 92]. Five $2 \times 2 \times 2$ cubes with different colors were generated and placed at different places in 3D space, then the synthetic camera took pictures from different views. The resolution of the images is 512 by

512. Table 6.1 gives the summary of the test results for these synthetic images. In this table both the camera's real positions, orientations and their corresponding computed values are given, as well as the number of corners used to get each result. Basically for every image, more than 19 corners are used to compute the result. The detailed information for each situation, i.e. the 2D image coordinates and corresponding 3D world coordinates for the corners, are omitted.

From Table 6.1, we tried to improve the results for Figure 6.6 by eliminating those corners who bring the biggest errors to the linear equations in the least-squares computation, then recomputing the transformation matrix and the camera parameters. the results are shown In Table 6.2. The first row is the expected result, the second row was obtained by using all the corners in the image as in Table 6.1, from the third to the last row, given number of corners was eliminated from the original dataset (total 20 elements) according to the errors they brought, and the new results are shown.

For the case of Figure 6.6, we show the results when changed some values in the original dataset to simulate the situation that there are 3 mismatches in the dataset (total 20 elements), i.e. there are some 2D corners with the wrong 3D coordinates: for example, corner (293, 279)'s 3D coordinates should be (1, 1, -1) but as the corner is mismatched to another nearby corner in the 2D-2D feature correspondences, consequently it gets the wrong 3D coordinate (1, -1, -1). This also happens to two other corners. Table 6.3 shows the results in this situation, the first row is still the expected result, while the second row is

obtained by eliminating one corner from the initial dataset (total 20 elements 3 of which have wrong 3D coordinates), then 2, 3 and more corners are eliminated from the dataset. In this table, one sees that after the three mismatched corners are eliminated from the original dataset, the result becomes stable, and further elimination will not affect the result significantly.

From Table 6.2 and 6.3, we may say that the method used here to improve the result accuracy is useful when we have some mismatches in the dataset which produce very large errors to the linear equations. After eliminating them, the recomputed result will improve significantly and the result will move towards the expected result. But this method may not be of help when the dataset only has noise, but doesn't have mismatches.

In Table 6.4 we show the results when we used different corners combination from the original dataset to compute the results. The corners used were randomly selected from the original dataset. When omitting one or two corners from the original dataset, the results didn't change very much, but when only 6 corners were left, the results varied greatly. Therefore, although theoretically we can get an answer with 6 corners, in practical more than 6 corners should be used in order to get stable solutions. When the dataset doesn't have mismatches, the more corners are used, the more stable the result will be.

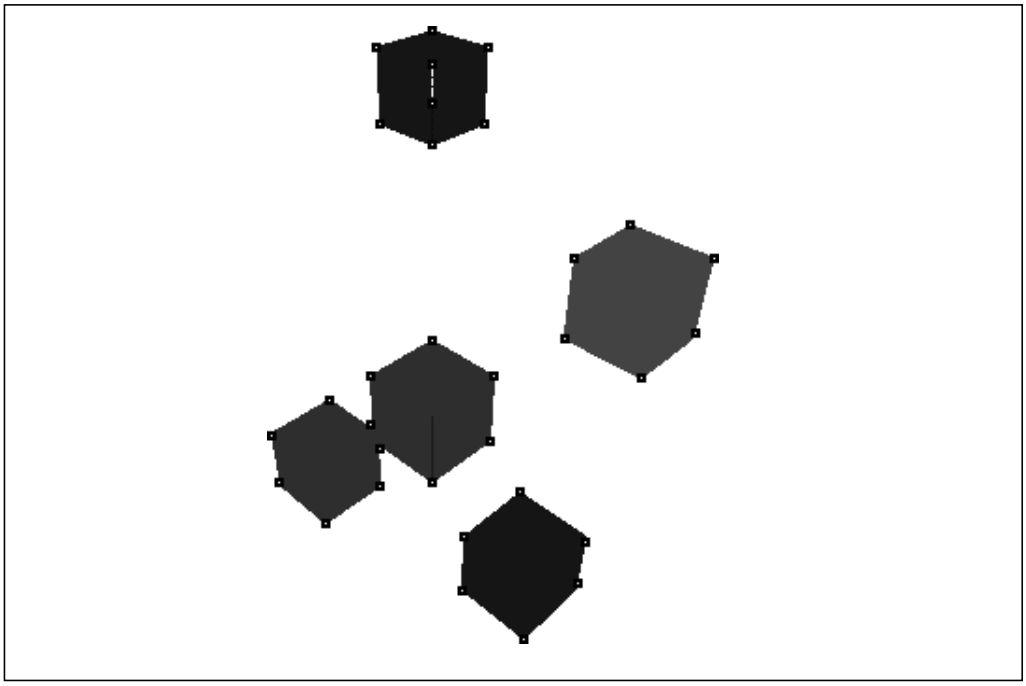

Figure 6.1 Synthetic Image 1 (with detected corners)

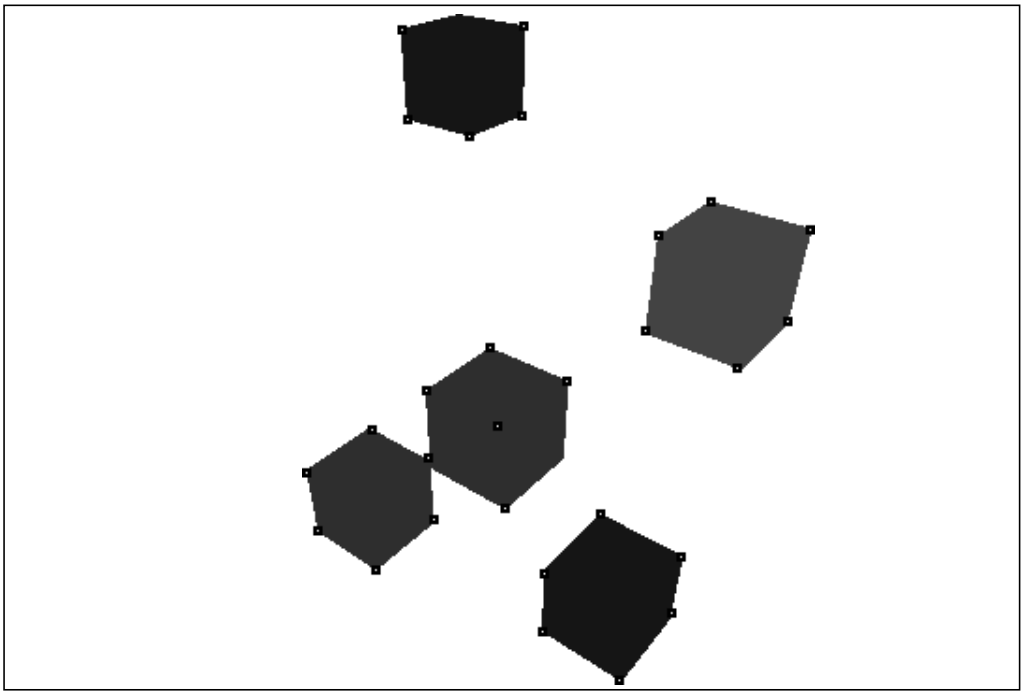

Figure 6.2 Synthetic Image 2 (with detected corners)

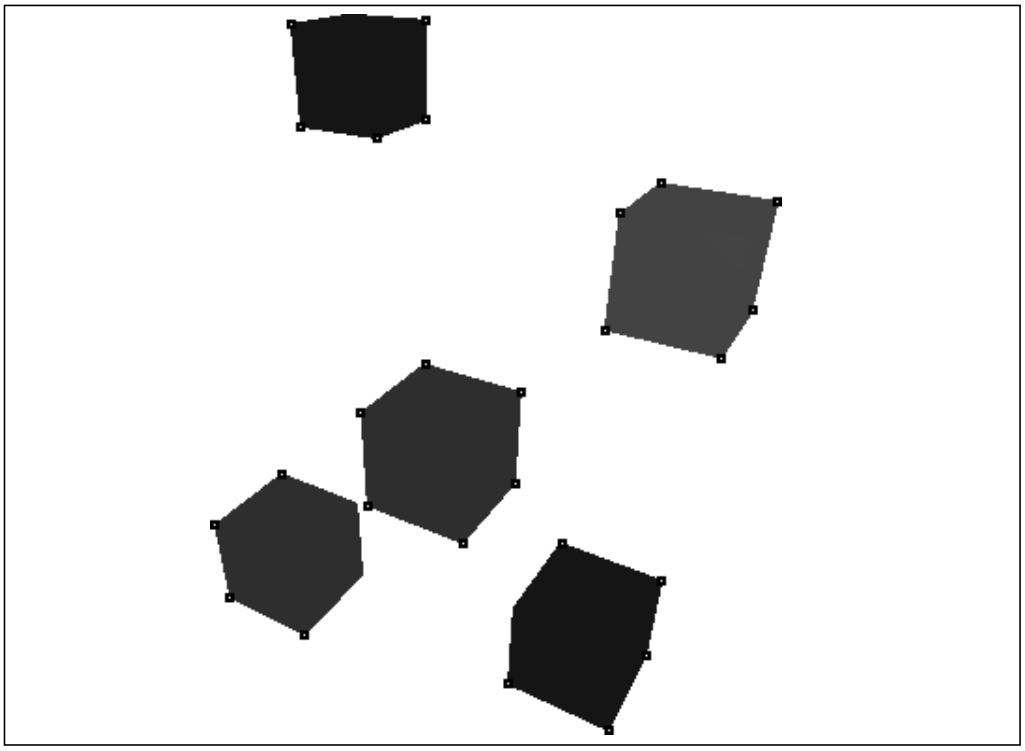

Figure 6.3 Synthetic Image 3 (with detected corners)

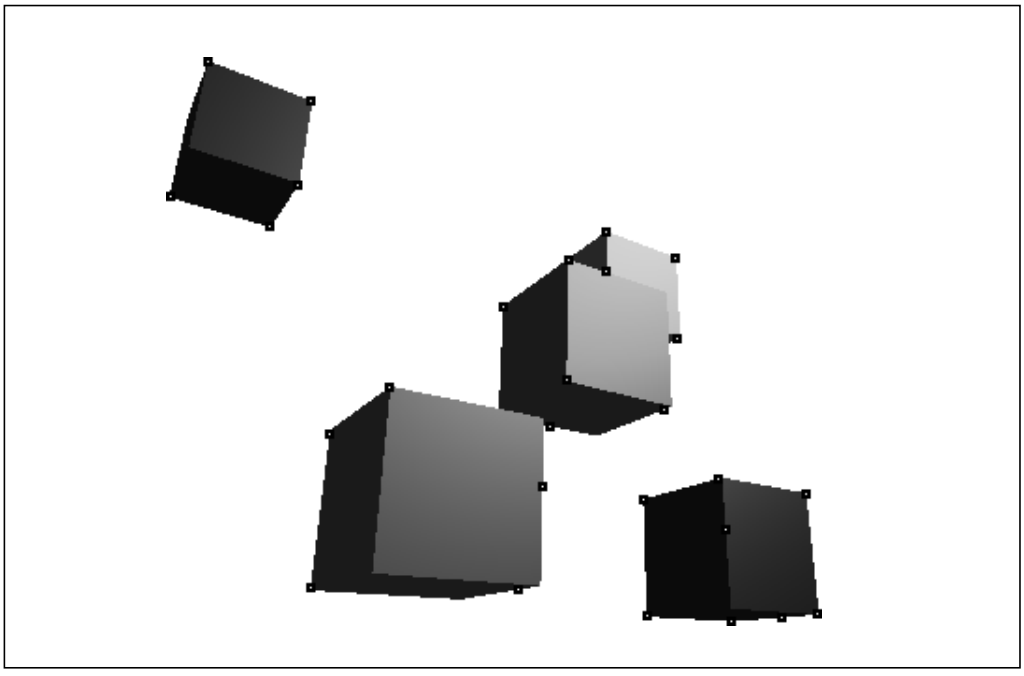

Figure 6.4 Synthetic Image 4 (with detected corners)

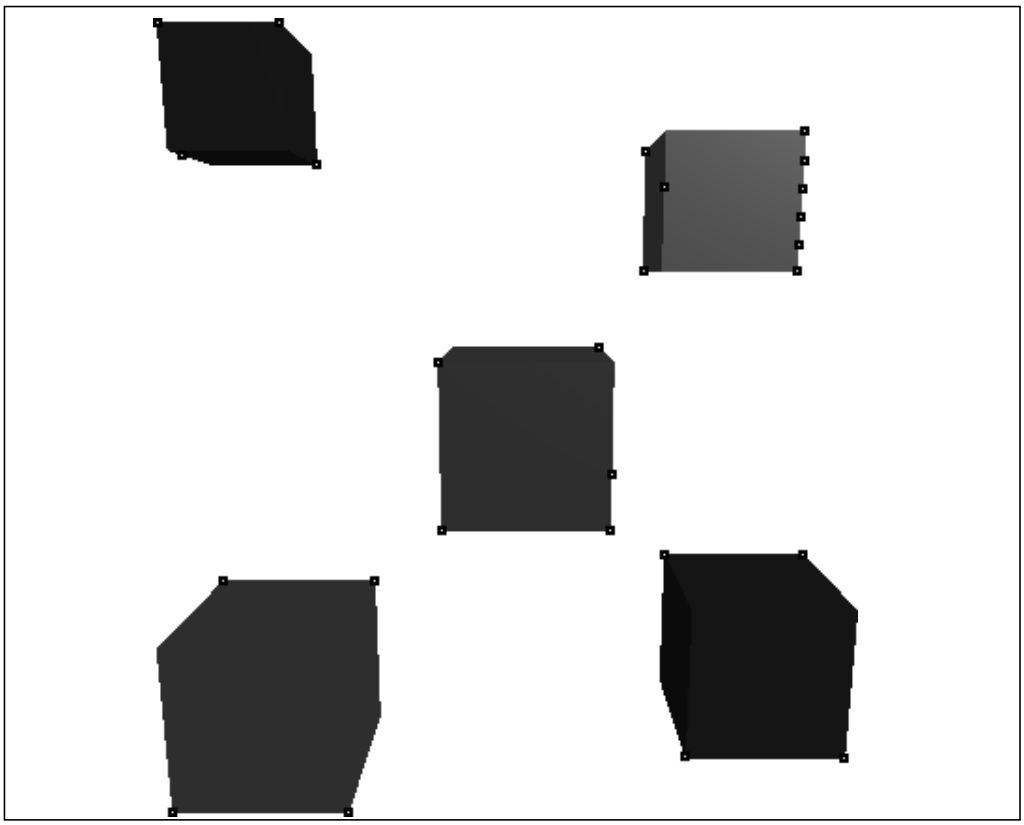

Figure 6.5 Synthetic Image 5 (with detected corners)

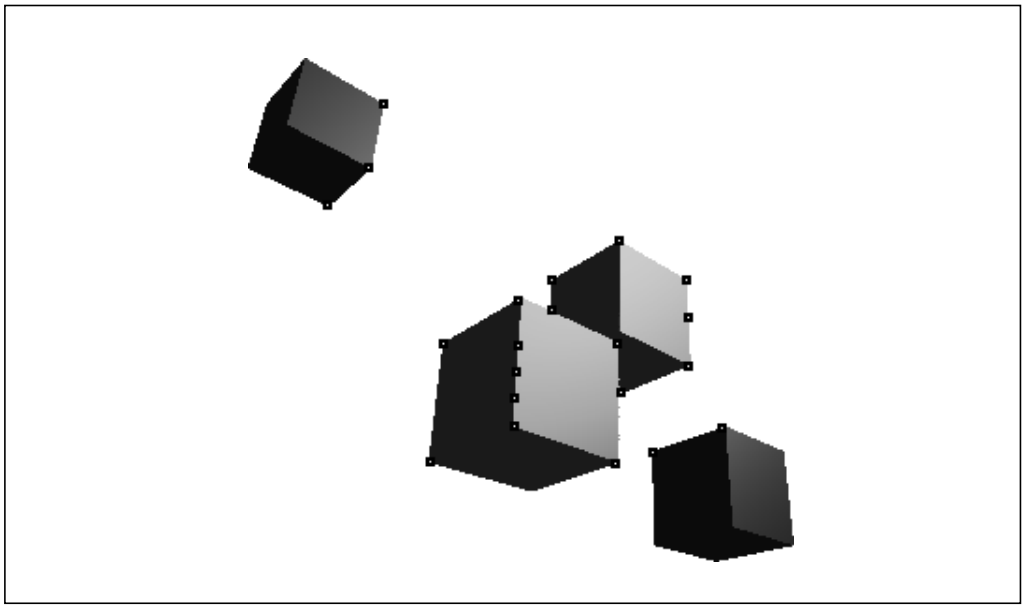

Figure 6.6 Synthetic Image 6 (with detected corners)

Table 6.1 Summary of the Results for Synthetic Images

Image Used	Results	Total Corners Used	Camera Position (x, y, z)			Camera Orientation (θ, ϕ, ψ) (in degrees)		
Figure 6.1	Expected	n/a	10	10	12	135	-40.3155	0
	Computed	24	10.0418	9.76797	12.3704	135.767	-40.2628	0.906025
Figure 6.2	Expected	n/a	10	8	10	128.66	37.9852	0
	Computed	24	10.1921	8.04824	10.087	129.241	-37.9178	0.44467
Figure 6.3	Expected	n/a	10	6	8	120.966	-34.4499	0
	Computed	26	10.138	5.98412	8.07359	120.983	-34.6151	0.089
Figure 6.4	Expected	n/a	10	-7	-5	55.0347	22.0712	0
	Computed	19	9.98793	-6.98462	-4.94183	55.3543	21.5163	-0.252
Figure 6.5	Expected	n/a	10	0	2	90	-11.3099	0
	Computed	19	10.4248	0.036744	2.08431	89.7911	-10.7542	0.03
Figure 6.6	Expected	n/a	10	-10	-8	45.4286	29.001	0
	Computed	20	10.6122	-10.4546	-8.25781	45.4537	29.3023	-0.31

Table 6.2 Improved Results for Dataset with only Noise

No. of Corners Eliminated	**Camera Position (x, y, z)**			**Camera Orientation** (θ, ϕ, ψ)		
n/a	**10**	**-10**	**-8**	**45.4286**	**29.001**	**0**
0	10.6122	-10.4546	-8.25781	45.4537	29.3023	-0.31
2	10.6533	-10.4718	-8.28137	45.0902	29.5975	-0.089
4	10.6665	-10.4804	-8.31937	44.1698	30.077	0.481594
6	10.7047	-10.5479	-8.35039	42.9077	31.0379	1.09046
8	10.7256	-10.5844	-8.33206	43.2051	31.1865	0.97729

Table 6.3 Improved Results for Dataset with Mismatches

No. of Corners Eliminated	**Camera Position (x, y, z)**			**Camera Orientation** (θ, ϕ, ψ)		
n/a	**10**	**-10**	**-8**	**45.4286**	**29.001**	**0**
0	5.91349	-6.8931	-5.4958	41.3477	26.4011	-0.396
1	6.0738	-7.27335	-4.77751	47.8276	22.3429	-3.481
2	7.15545	-8.27474	-6.21438	44.9449	25.4265	-1.402
3	10.6045	-10.4591	-8.25704	45.5382	29.2086	-0.377
4	10.6188	-10.477	-8.26687	45.4569	29.4223	-0.304
5	10.6538	-10.4835	-8.29198	45.1635	29.5445	-0.143

Table 6.4 Results for different Corners Combination

No. of Corners Used	**Camera Position (x, y, z)**						**Camera Orientation** (θ, ϕ, ψ)		
n/a	**10**	**x diff**	**-7**	**y diff**	**-5**	**z diff**	**55.0347**	**22.0712**	**0**
19	9.98793	0.01207	-6.98462	-0.01538	-4.94183	-0.05817	55.3543	21.5163	-0.252
18	9.98718	0.01282	-6.98176	-0.01824	-4.9387	-0.0613	55.2745	21.591	-0.233
18	10.0584	-0.0584	-7.03616	0.03616	-4.99182	-0.00818	54.9276	21.9051	-0.072
18	10.0036	-0.0036	-6.9943	-0.0057	-4.93841	-0.06159	55.325	21.5536	-0.254
18	10.0079	-0.0079	-6.98864	-0.01136	-4.92471	-0.07529	55.417	21.4322	-0.289
17	9.9357	0.0643	-6.96312	-0.03688	-4.90574	-0.09426	55.0806	21.6405	-0.137
8	10.1605	-0.1605	-7.04325	0.04325	-5.06112	0.06112	55.4006	21.6337	-0.126
6	10.1944	-0.1944	-6.986	-0.014	-5.13373	0.13373	53.201	22.3552	1.18394
6	8.62728	1.37272	-6.44897	-0.55103	-4.46529	-0.53471	56.2818	15.1994	-0.045
6	11.0048	-1.0048	-7.41336	0.41336	-5.48329	0.58329	64.0193	12.1546	-5.604

6.2 Images taken with a real camera

The experimental environment was set up in the Laboratory of Computational Intelligence of University of British Columbia. The “ACME” group provide software to control the camera, test station and to capture the image. A SONY 3CCD Color Video Camera (model DXC-950) was mounted on the optical table and the objects are put on the test station which can move and rotate under program control. The test station initially was moved to a fixed position and only rotated between images. The origin of the World Coordinate System was set to the center of the test station, the Y axis was pointing to the camera (may not align with camera optical axis), the Z axis was perpendicular to surface of the optical table, with a right-hand coordinate system was used, which determined the X axis uniquely. Figure 6.7 is a picture of the environment.

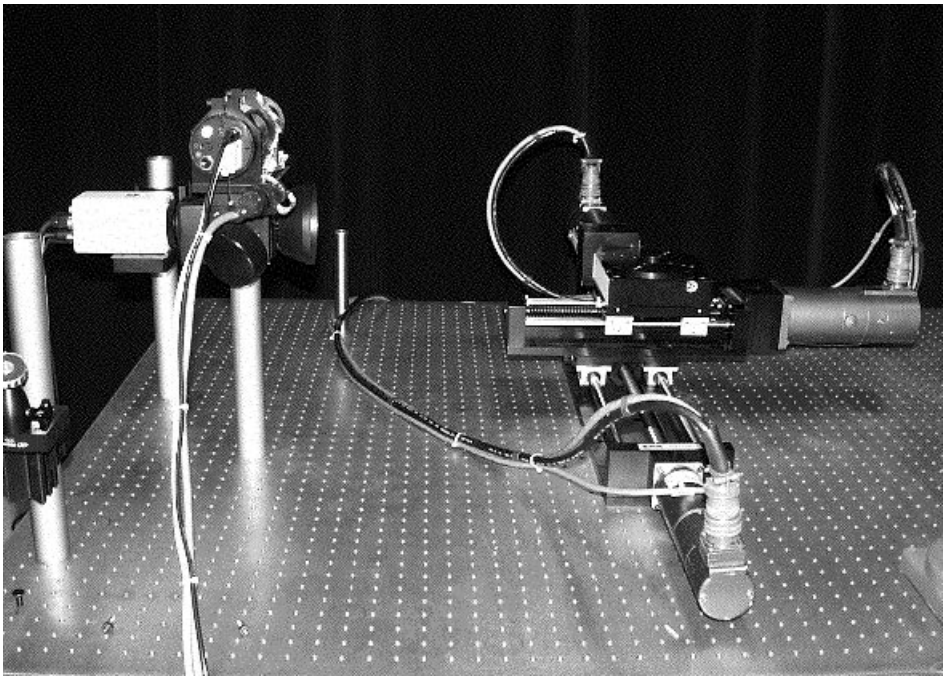

Figure 6.7 the Optical Table, the Camera and the Test Station

The set up of the test station: First, a $13\ \text{cm} \times 15.6\ \text{cm} \times 1.6\ \text{cm}$ raiser board was used, and a coordinate system was built on it. With the origin in the center and in the bottom plane of the board, X,Y axis are shown Figure 6.8 and Z points away from the board plane; Second, we placed several objects on the paper, and measured the coordinates for the objects in this coordinate system. As we knew the size of each object, we could get the coordinate for the corners on the objects in this coordinate system. Third, we put the board with the objects onto the test station, let the X,Y axes on the papers aligned to X_W, Y_W axes of the World Coordinates System, and made the origin of the both coordinate systems

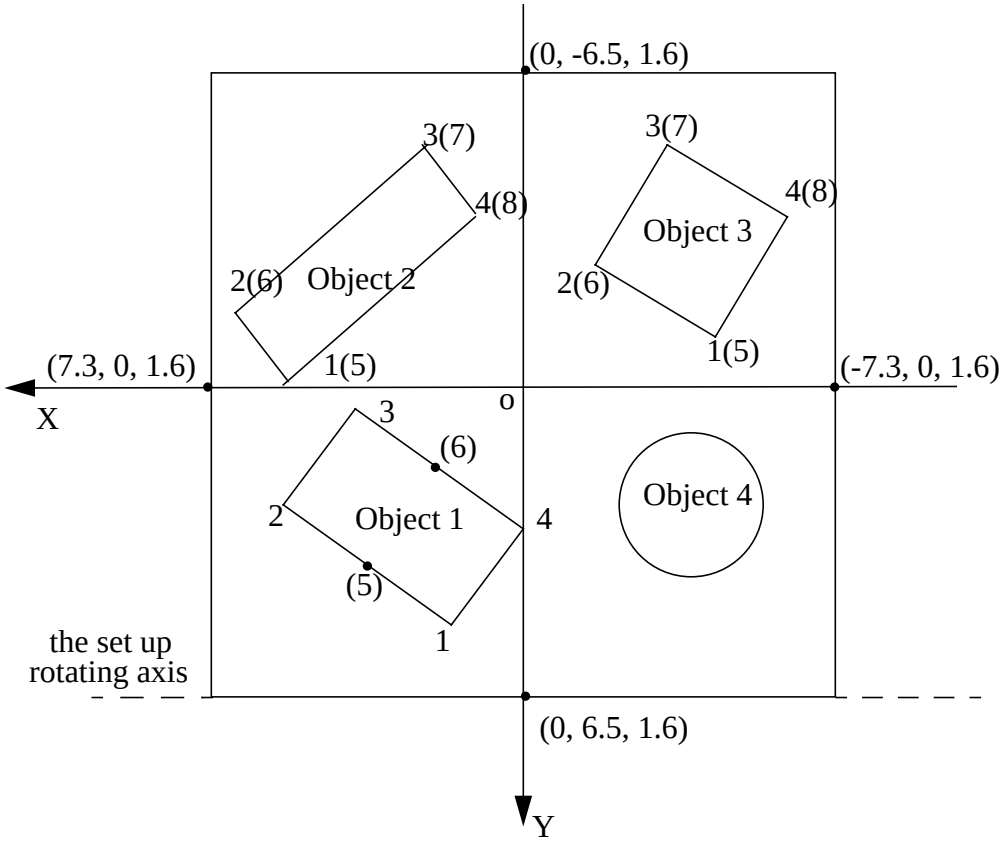

Figure 6.8 Top View of the Set Up (before Rotating)

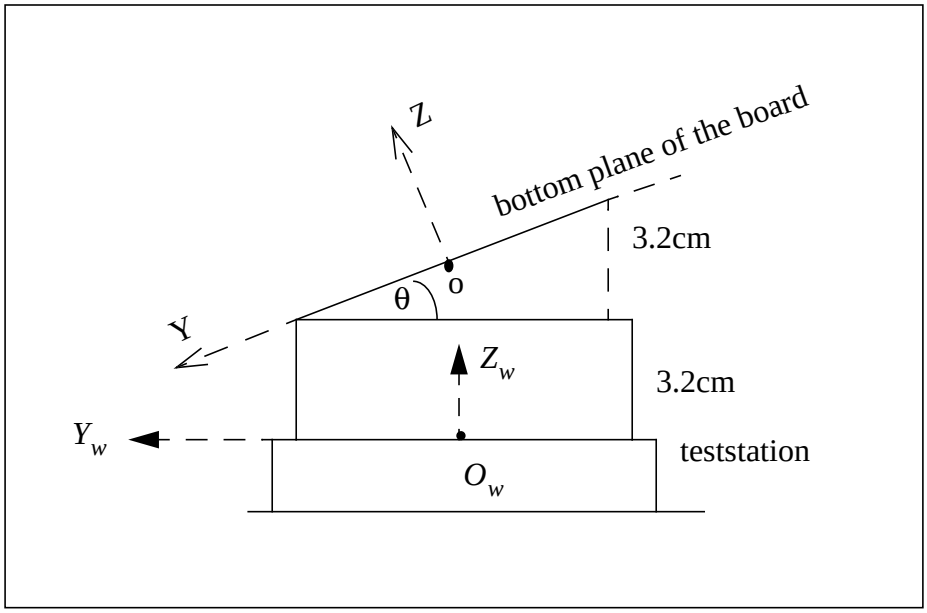

Figure 6.9 Side View of the Set Up (after rotating)

be at the same place. Next the board was lifted up 3.2cm in z direction. Finally, the board was rotated around an axis parallel to the X axis as shown in Figure 6.7 to make the board tilt. The result is shown in Figure 6.9.

The camera focus length is set to 0.02, and zoom is set to 85. Twelve images of the objects on the teststation were captured with a resolution of 640×480. The motion between each pair of two views was a rotation of 10 degrees of the test station. This gives us the same effect as a rotation of 10 degrees of the camera while all the objects are still. Figure 6.10 shows the initial position. We took a total of 6 images when test station rotated counter-clockwise, then we set the test station to the initial position, and took 6 more images as the test station rotated clockwise. These images were not corrected for lens distortion.

In this paper, only the test results of the 6 images where the test station rotated counter-clockwise are given in Table 6.5 - Table 6.16.

In Table 6.5 - 6.16, the 3D world coordinates of the corners were obtained by transforming from the coordinates in the board's coordinate system $(x\ y\ z)$ to the World Coordinate System $(X_w\ Y_w\ Z_w)$ by two steps. First the $y-z$ axes was rotated around x by θ (from y to z) to make y parallel to Y_w and z parallel to Z_w, then the origin o was moved to the new place O_w.

In Table 6.5 - 6.16, we use the notation $m{:}n$ to refer to the number n corner in the object m, so we can identify each corner uniquely. Figure 6.8 should be used as a reference, as it shows the number for each corner clearly. The number between the parentheses means this corner is at the top of the objects. With the corners are numbered, we can have a better understanding for each result - how many corners and which corners are used to get it, and we can compare different results more clearly. The correspondences of the 2D image coordinates, and their 3D coordinates of corners are also shown in tables.

In Table 6.6 - 6.16, "Distance" is the distance from the computed camera position to the origin of the World Coordinate System. Ideally it should be a constant value, as we only had rotation in the experiment, the distance between the camera and the origin of the World Coordinate System didn't change. "Rotation" is the angle between the computed camera position and the Y axis, ideally this angle minus the computed angle for the initial

position (Position 0) should equal to the rotation degree of the objects. “w.o.” means “without”.

As Table 6.6 - 6.16 shows, the value of “Distance” varies significantly, although we know it should be a constant, while the “Rotation” angles are relatively stable, and close to the true values for every image. The results are quite promising, although they need a lot of improvements.

There are several reasons which may bring errors to the results:

Lack of corners: In every case, less than ten corners are used, and as we can see from the results for the synthetic images, when less than 10 corners are used, the camera positions varies a lot, sometimes very far from the true values. As in these cases, the dataset doesn’t have mismatches, only have noise, the method we used to improve the accuracy in synthetic images will not be helpful here.

Large noise in the dataset: for every image, the errors in the linear equations are very large, this means there is large noise in the dataset. This noise may result from the inaccuracy of the environment set up, the inaccuracy measuring the objects and the inaccuracy of the equipment. Also the corners in the real world are rather blunt, not as expected defined by sharp edges, and the detected corners will not be localized very well. The small difference in the corner localizations may result in big difference in the Trans-

formation Matrix values, therefore the camera parameters which are solely based on discomposing Transformation Matrix will change significantly.

To increase the number of useful corners. We could use the following methods:

1) From Figure 6.10 - 6.15, we can get plenty of corners, although we only use less than 10 now. Among these corners, some of them are caused by the texture changes on the objects, these corners are quite stable and can be traced from one image to another, therefore, if their 3D coordinates are available, they can be used to compute results as well. This will increase the number of useful corners.

2) Another method of increasing the number of corners is to reduce threshold, we will get more useful corners, although at the same time we also get more false corners. This method has side effect. With more false corners, the number of mismatches will also increase, the method to eliminate mismatches must be improved, otherwise human interference will be increased.

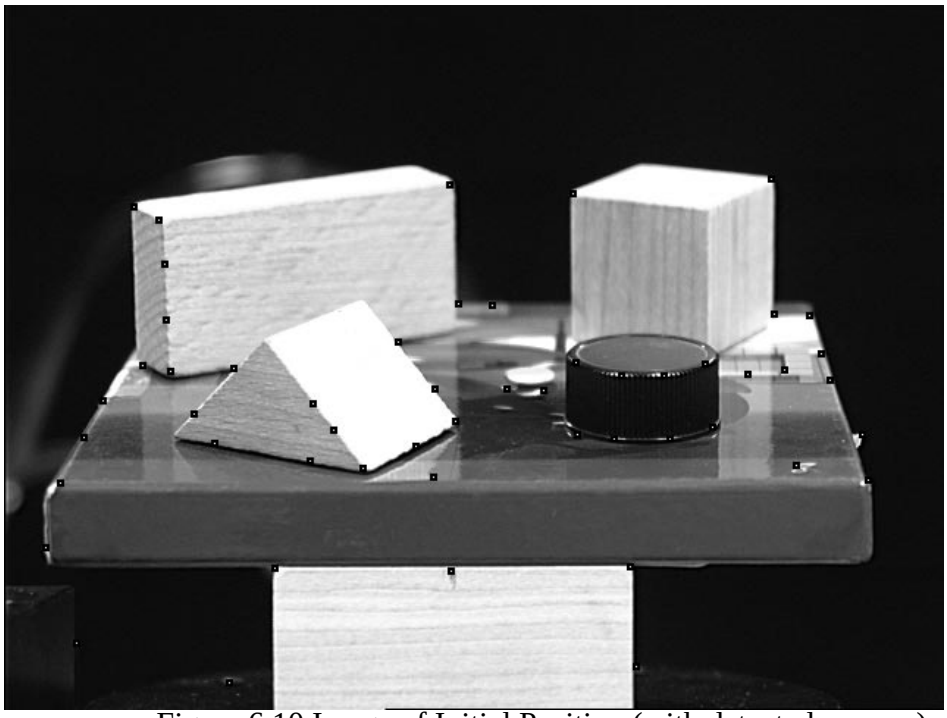

Figure 6.10 Image of Initial Position (with detected corners)

Table 6.5 3D-2D Corner Correspondences for Rotating 0 Degree (Initial Position)

Corners	**World Coordinates**			**Image Coordinates**	
1:1	1.4	6.015	3.323	244.0	315.0
1:3	0	3.108	4.062	306.9	283.1
2:1	6	0.200	4.800	113.0	249.0
2:5	6	0.985	7.892	105.0	146.0
2:6	6.9	-0.372	8.236	88.0	137.0
2:8	0.5	-2.698	8.827	303.0	122.0
3:5	-4.75	-0.471	8.251	479.6	140.5
3:6	-2	-1.92492	8.621	386.9	128.0
3:8	-6.3	-3.088	8.916	522.0	118.9.0

Table 6.6 Results for Rotating 0 Degree (Initial Position)

Points Used	Camera Position			Camera Orientation			Distance	Rotation
9	-1.19153	35.7931	6.56891	-176.083	-19.183	358.797	36.4104	1.90664
8 w.o. 1:1	-0.71776	33.0026	7.1286	-176.47	-19.4893	358.655	33.7713	1.24591
8 w.o. 1:4	1.2852	32.2614	6.70073	-177.927	-39.3094	357.331	32.975	-2.28129
8 w.o. 2:1	-2.18205	24.5159	6.73176	-174.402	-45.0536	359.765	25.5168	5.08624
8 w.o. 2:5	-0.987056	38.415	6.79826	-175.78	-18.5052	358.597	39.0244	1.47187
8 w.o. 2:6	-0.940546	36.3632	6.76213	-178.335	-24.9673	358.344	36.9986	1.48164
8 w.o. 2:8	-1.46857	36.5559	6.82802	-179.504	-18.315	358.205	37.2171	2.30052
8 w.o. 3:5	-0.659698	30.5111	5.88835	-174.739	-18.3968	359.017	31.0811	1.23863
8 w.o. 3:6	-1.48539	36.3099	6.46822	-173.818	-15.1544	359.356	36.9114	2.34259
8 w.o. 3:8	-3.97321	51.4027	6.69851	-166.215	2.49555	359.858	51.9894	4.41993
7 w.o. 2:6,8	-1.00061	37.9319	7.2265	177.615	-25.2165	357.296	38.6271	1.51106
7 w.o. 1:4 3:6	0.495826	33.1539	6.63077	-175.897	-35.4005	358.111	33.8141	-0.856811
6 w.o. 1:1 2:1,8	0.164975	10.8657	5.44271	-179.844	-75.5728	357.709	12.1538	-0.869861
6 w.o. 2:1,5 3:8	1.47907	-79.4715	4.92696	-1.97711	80.867	2.844	79.6378	-1.06623
Average	Note: the average for Distance and Rotation is based on 6+ points cases						**36.1947**	**1.6556**

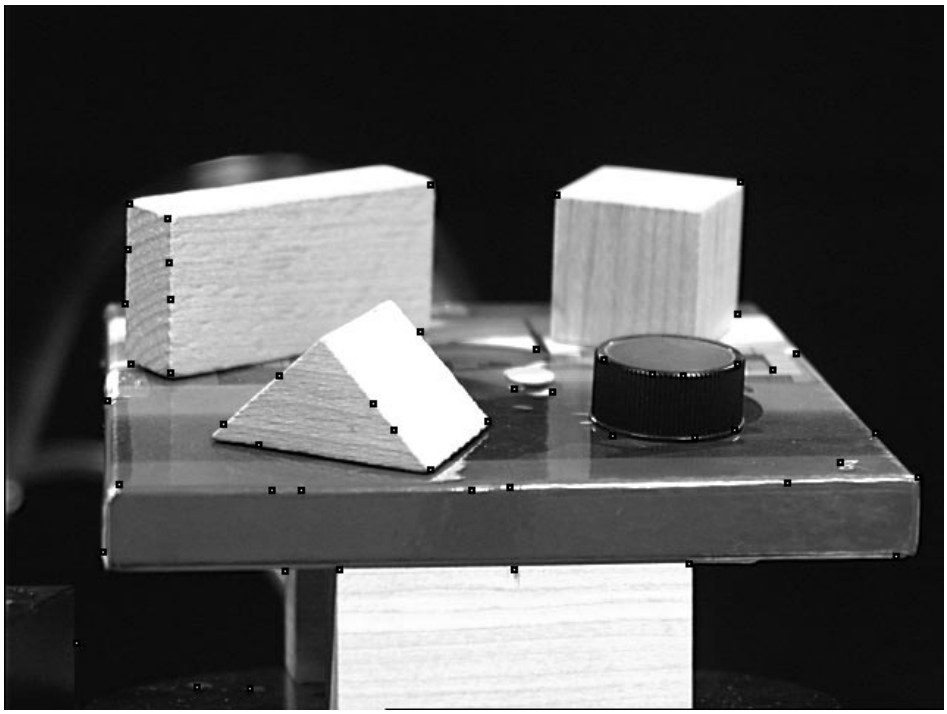

Figure 6.11 Image of Rotating 10 Degrees (with detected corners)

Table 6.7 3D-2D Corner Correspondences for Rotating 10 Degrees (Position 1)

Corners	**World Coordinates**			**Image Coordinates**	
1:4	0	3.10769	4.06154	329	283
1:6	2	2.58062	6.45492	262	201
2:1	6	0.2	4.8	112.6	250.7
2:5	6	0.985231	7.89185	111.4	144.6
2:6	6.9	-0.37169	8.23646	85	135.1
2:8	0.5	-2.69785	8.82723	290	122
3:6	-2	-1.92492	8.6062	376	129
3:8	-6.3	-3.088	8.916	501	120.9

Table 6.8 Results for Rotating 10 Degrees (Position 1)

Points Used (Total)	**Camera Position**			**Camera Orientation**			**Distance**	**Rotation**
8	5.48905	22.9722	6.7204	179.869	-29.415	359.35	24.5564	13.4385
8 w.o. 1:4	4.68444	17.9718	6.88622	-176.784	-25.2832	359.209	19.8078	14.6094
7 w.o. 1:6	4.98435	23.9106	6.1267	-178.998	-23.1697	359.41	25.1813	11.7751
7 w.o. 2:1	3.18449	16.6761	7.21423	-177.17	-43.4283	2.365	18.4466	10.8111
7 w.o. 2:5	7.73813	36.9259	7.31632	175.79	-24.6311	358.45	38.4308	11.8355
7 w.o. 2:6	6.19194	26.8771	6.90123	176.009	-33.8493	359.057	28.4314	12.9734
7 w.o. 2:8	5.34249	23.2807	6.72819	178.764	-28.0334	358.954	24.8154	12.9245
7 w.o. 3:6	5.44111	22.5071	6.70864	-179.969	-29.4869	359.388	24.1077	13.5905
7 w.o. 3:8	5.43493	22.4029	6.70917	178.264	-28.6728	358.835	24.0092	13.6365
6 w.o. 2:5 3:8	10.7017	58.1264	7.54251	176.195	-22.805	358.585	59.5827	10.432
6 w.o. 2:5 1:4	7.6171	24.178	6.37251	177.856	-56.0349	359.922	26.1382	17.4866
6 w.o. 1:6 2:1	0.468456	11.1364	5.90769	-178.94	-71.3767	5.733	12.6151	2.40874
6 w.o. 2:6 2:1	6.12774	26.4331	6.89288	176.283	-33.7153	359.126	27.9959	13.0518
Average	Note: The average for Distance and Rotation is based on 6+ points cases						**25.3097**	**12.8438**

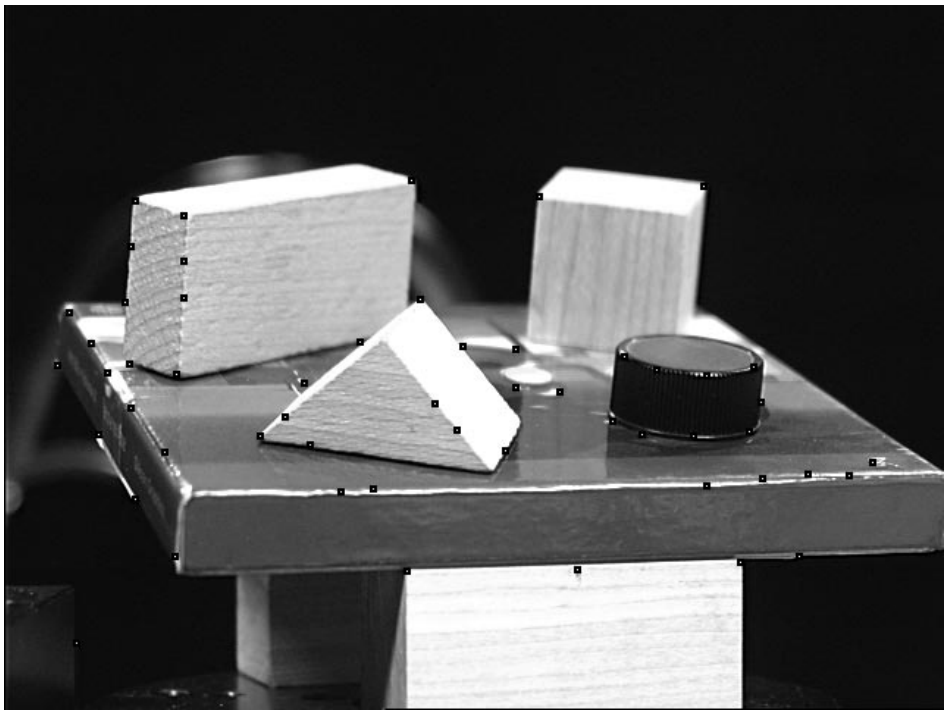

Figure 6.12 Image of Rotating 20 Degrees (with detected corners)

Table 6.9 3D-2D Corner Correspondences for Rotating 20 Degrees (Position 2)

Corners	**World Coordinates**			**Image Coordinates**	
1:1	1.4	6.01538	3.32308	332	316
1:3	5.4	3.98	3.84	173.8	293
1:6	2	2.58062	6.45492	283	200
2:1	6	0.2	4.8	117	250.3
2:5	6	0.985231	7.89185	122.4	142.6
2:6	6.9	-0.371692	8.23646	89	133.2
2:8	0.5	-2.69785	8.82723	277	119
3:2	-2	-1.92492	8.62062	364	130
3:8	-6.3	-3.088	8.916	475.1	123

Table 6.10 Results for Rotating 20 Degrees (Position 2)

Points Used (Total)	**Camera Position**			**Camera Orientation**			**Distance**	**Rotation**
9	11.3026	26.714	7.02832	170.564	-47.2411	0.593	29.846	22.9331
8 w.o. 1:1	11.2822	27.0695	7.16351	171.208	-43.5675	0.549	30.1888	22.6257
8 w.o. 1:2	9.62931	21.5997	6.78476	174.635	-44.0568	1.325	24.6029	24.0277
8 w.o.1:4	9.90793	21.0632	7.11766	172.791	-51.387	1.915	24.341	25.1918
8 w.o. 2:1	6.84421	18.4809	6.77531	176.85	-49.2261	3.708	20.8397	20.3216
8 w.o. 2:5	20.9982	46.5246	7.60667	160.079	-37.7731	356.138	51.6074	24.2913
8 w.o. 2:6	14.1973	31.4147	7.14322	160.08	-49.6937	356.733	35.2061	24.3197
8 w.o. 2:8	9.58789	22.4348	7.02396	175.821	-49.6601	3.036	25.3887	23.1403
8 w.o. 3:6	11.2932	26.6809	7.03666	170.771	-47.085	0.731	29.8148	22.9414
8 w.o. 3:8	11.4537	27.5082	7.0053	170.444	-47.0506	0.464	30.6098	22.6056
7 w.o. 3:6,8	10.7361	25.7997	7.1574	173.06	-47.988	1.859	28.8464	22.5938
7 w.o. 1:2 3:8	9.67261	21.8365	6.7807	174.567	-44.0018	1.269	24.8268	23.8912
7 w.o. 1:2 2:8	8.28418	17.0642	6.62905	179.496	-45.7527	3.148	20.0938	25.8952
7 w.o. 2:1 2:8	6.21391	16.7014	6.34874	178.128	-48.9559	4.35516	18.9171	20.4082
7 w.o. 2:6 3:6	14.036	30.9422	7.14252	160.338	-49.8076	356.886	34.7195	24.4
Average	Note: The average for Distance and Rotation is based on 6+ points cases						**28.6566**	**23.3055**

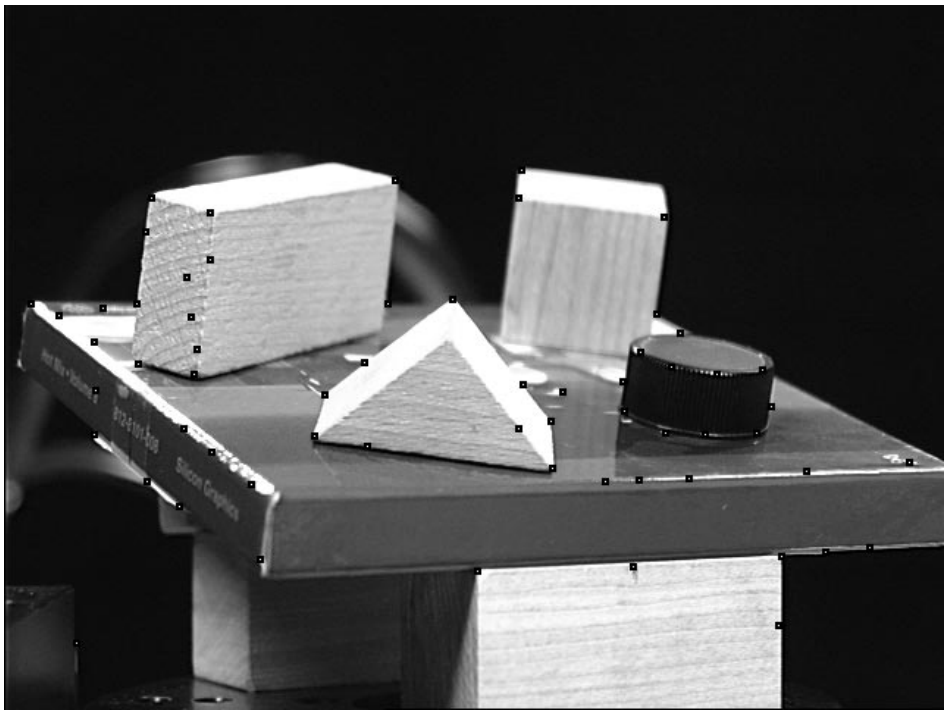

Figure 6.13 Image of Rotating 30 Degrees (with detected corners)

Table 6.11 3D-2D Corner Correspondences for Rotating 30 Degrees (Position 3)

Corners	**World Coordinates**			**Image Coordinates**	
1:1	1.4	6.01538	3.32308	373	315.1
1:2	5.4	3.98	3.84	210.9	293.1
1:4	0	3.10769	4.06154	372.5	283.8
1:6	2	2.58062	6.4592	305	200.1
2:1	6	0.2	4.8	128.9	251
2:5	6	0.985231	7.89185	140.5	140.6
2:6	6.9	-0.371692	8.23646	100	131.2
2:8	0.5	-2.69785	8.82723	265.1	119
3:5	-4.75	-0.471077	8.25138	449	144.3
3:6	-2	-1.92492	8.62062	350	131.1

Table 6.12 Results for Rotating 30 Degrees (Position 3)

Points Used (Total)	Camera Position			Camera Orientation			Distance	Rotation
10	29.1934	53.493	7.28785	142.003	-8.19495	357.267	61.3748	28.6232
9 w.o. 1:1	20.7204	35.2711	6.96452	155.088	-12.812	358.04	41.4957	30.4326
9 w.o. 1:2	26.2345	48.4385	6.97719	146.452	-6.09064	357.66	55.5267	28.4402
9 w.o.1:4	17.1728	26.4651	6.89428	156.767	-43.1004	358.643	32.293	32.9789
9 w.o. 1:6	64.4524	76.104	4.88387	132.016	44.9476	187.651	99.8488	40.2612
9 w.o. 2:1	15.7803	34.3523	6.70741	161.227	-35.5572	181.579	38.3939	24.6725
9 w.o. 2:5	72.9265	69.6617	-3.32585	110.406	41.3708	192.467	100.906	46.3117
9 w.o. 2:6	29.9922	51.7653	9.82063	128.323	-21.1107	353.497	60.6269	30.0875
9 w.o. 2:8	26.8773	49.6762	7.152	142.376	-8.05692	357.35	56.9321	28.4156
9 w.o. 3:5	30.5821	54.9613	6.84871	147.469	-16.1866	357.091	63.2686	29.0929
9 w.o. 3:6	30.1285	55.5235	7.48369	140.808	-6.57456	357.36	63.6128	28.4854
8 w.o. 1:2 2:8	20.6229	38.9202	6.64157	149.338	-5.41306	357.879	44.5443	27.9181
8 w.o. 1:1 1:4	18.1449	28.5881	6.87919	156.2	-36.0127	358.311	34.552	32.4033
8 w.o. 2:8 3:6	28.6239	54.6743	7.50599	134.816	-1.39268	357.86	62.1687	27.6336
7 w.o. 1:6 3:5,6	97.9398	62.5918	12.0497	119.11	58.4229	195.189	116.855	57.418
7 w.o. 2:1 5:8	33.1988	63.1305	6.61737	136.015	2.32176	358.973	71.6339	27.7388
7 w.o. 1:2,4 2:1	5.96907	13.6676	7.87541	172.747	-52.9939	186.737	16.8658	23.5924
Average	Note: The average for Distance and Rotation is based on 6+ points cases						**60.0529**	**32.0303**

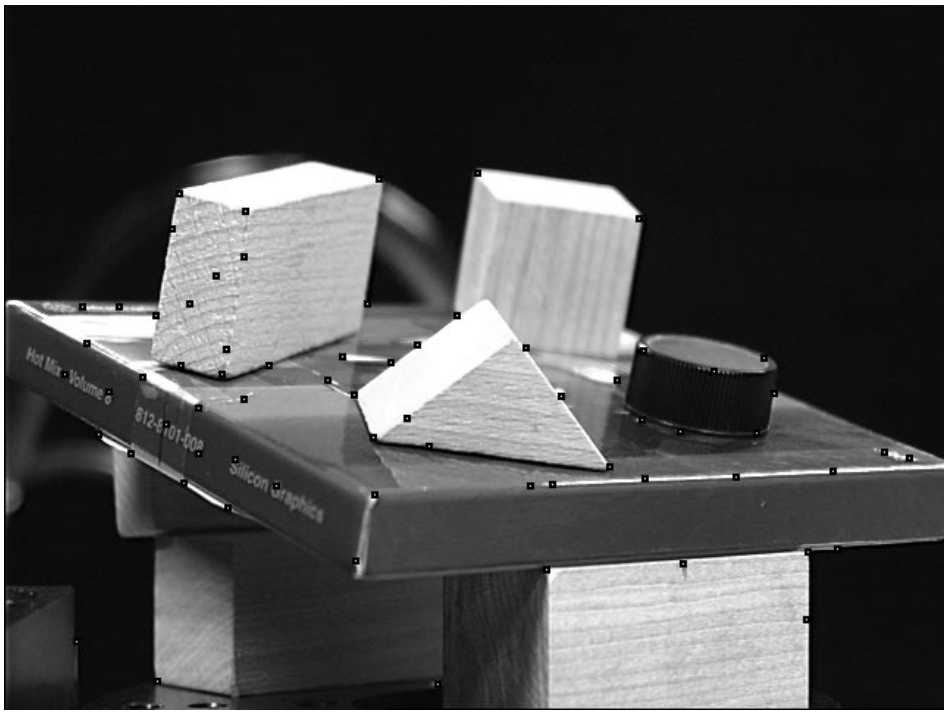

Figure 6.14 Image of Rotating 40 Degrees (with detected corners)

Table 6.13 3D-2D Correspondences for Rotating 40 Degrees (Position 4)

Corners	**World Coordinates**			**Image Coordinates**	
1:1	1.4	6.01538	3.32308	412.1	314
1:2	5.4	3.98	3.84	251	294
1:3	3.8	1.16923	4.55385	238	265
1:6	2	2.58062	6.45492	328	200
2:1	6	0.2	4.8	148	251.1
2:5	6	0.985231	7.89185	164.4	139.7
2:6	6.9	-0.371692	8.23646	119.1	127.1
2:8	0.5	-2.69785	8.82723	255	118
3:5	-4.75	-0.471077	8.25138	432	145.1
3:7	-3.4	-4.63877	9.30985	322	114

Table 6.14 Results for Rotating 40 Degrees (Position 4)

Points Used (Total)	Camera Position			Camera Orientation			Distance	Rotation
10	26.8763	31.1218	6.46706	138.546	-29.057	356.316	41.626	40.8134
9 w.o. 1:1	26.743	32.7198	6.57235	140.484	-16.3663	357.255	42.7664	39.2603
9 w.o.1:2	24.9866	29.2426	6.4656	141.339	-29.3495	356.87	39.0034	40.5125
9 w.o. 1:3	24.1743	27.9653	6.70251	141.991	-31.8619	356.749	37.5683	40.8414
9 w.o. 1:6	64.5819	52.0711	3.84704	120.355	46.667	9.16	83.0483	51.1215
9 w.o. 2:1	18.771	25.4901	6.64558	153.624	-43.3877	1.958	32.3459	36.3681
9 w.o. 2:5	36.059	42.4532	8.16764	124.167	-10.3921	354.801	56.296	40.344
9 w.o. 2:6	24.5228	26.2293	6.94364	130.412	-36.9868	353.698	36.5726	43.0742
9 w.o.2:8	23.1316	26.4822	6.65169	138.858	-28.5895	356.04	35.7858	41.1365
9 w.o. 3:5	28.1108	38.1176	6.31312	151.891	-43.9295	0.413	47.781	36.4079
9 w.o. 3:7	24.145	27.6109	6.59361	145.377	-38.4359	358.206	37.2669	41.1688
8 w.o. 1:2 3:7	23.0461	26.4532	6.64641	146.466	-38.1165	358.465	35.7081	41.0625
8 w.o. 1:3 2:8	20.5098	23.2878	6.91983	141.877	-31.1129	356.25	31.794	41.3707
8 w.o. 2:6 3:5	32.805	33.3171	6.18469	128.713	-45.1882	352.018	47.164	44.5563
Average	Note: The average for the Distance and Rotation is based on 6+ points cases						**43.1948**	**41.2884**

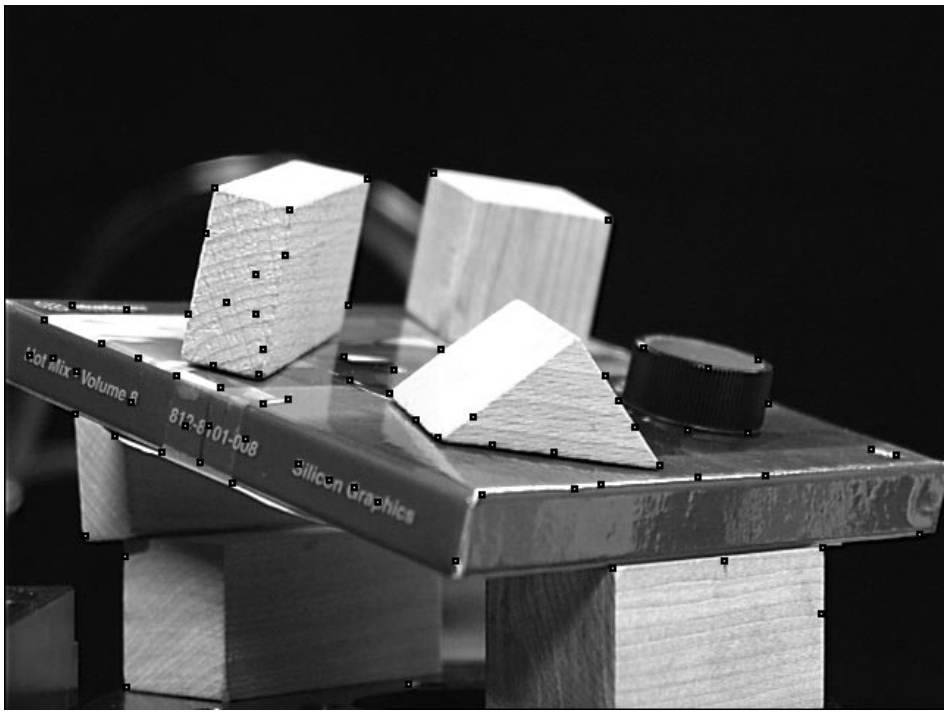

Figure 6. 15 Image for Rotating 50 Degrees (with detected corners)

Table 6.15 3D-2D Correspondences for Rotating 50 Degrees (Position 5)

Corners	**World Coordinates**			**Image Coordinates**	
1:1	1.4	6.01538	3.32308	446	313.3
1:2	5.4	3.98	3.84	295	293.9
1:3	3.8	1.16923	4.55385	262	264
1:6	2	2.58062	6.45492	349	200
2:1	6	0.2	4.8	173	251.1
2:5	6	0.985231	7.89185	194	139
2:6	6.9	-0.371692	8.23646	142.9	123.5
2:8	0.5	-2.69785	8.82723	247	118
3:5	-4.75	-0.471077	8.25138	411.1	146.1
3:7	-3.4	-4.63877	9.30985	291.9	114

Table 6.16 Results for Rotating 50 Degrees (Position 5)

Points Used (Total)	Camera Position			Camera Orientation			Distance	Rotation
10	28.0929	23.7668	6.49889	130.911	-29.0166	356.821	37.3672	49.7685
9 w.o. 1:1	28.8184	25.3475	6.53445	133.783	-13.2147	357.778	38.9319	48.6664
9 w.o. 1:1	26.5435	22.5195	6.58384	132.983	-30.31	357.218	35.4264	49.6887
9 w.o.1:2	24.2811	20.6912	6.87607	137.599	-33.9252	358.004	32.634	49.5639
9 w.o. 1:3	61.6795	26.2833	3.51029	100.381	46.2668	13.377	67.1379	66.9198
9 w.o. 1:6	18.1432	20.0653	7.30474	151.963	-46.389	5.469	28.0205	42.1201
9 w.o. 2:1	33.6627	27.6114	7.14702	119.098	-16.3233	354.65	44.1208	50.6401
9 w.o. 2:5	24.0606	19.776	6.83853	126.185	-36.0853	355.332	31.8868	50.5824
9 w.o. 2:6	25.9547	21.8486	6.60974	131.524	-28.5004	356.748	34.5644	49.9094
9 w.o.2:8	36.9139	31.6836	5.83267	129.857	-34.6381	356.299	48.995	49.3602
9 w.o. 3:5	28.3515	23.7697	6.56202	133.061	-33.5499	357.332	37.5748	50.0238
9 w.o. 3:7	27.3021	22.8648	6.66096	133.531	-33.6171	357.385	36.2294	50.0547
8 w.o. 1:2 3:7	130.168	-47.8643	41.0616	66.4242	62.8065	33.961	144.64	69.8109
8 w.o. 1:3 2:8	84.0443	12.664	9.34882	91.1564	59.7362	20.416	85.5057	81.431
8 w.o. 2:6 3:5	20.9232	18.145	6.8126	135.15	-45.4687	359.145	28.5207	49.0675
Average	Note: The average for Distance and Rotation is based on 6+ points cases						**48.7704**	**53.8405**

Chapter 7

Conclusions and Future Work

7.1 Conclusions

To merge Computer Generated Images with Real Video Images seamlessly in real time, it is necessary to recover the real camera parameters, including extrinsic parameters (position, orientation) and intrinsic parameters. This thesis proposes and analyzes a system which can be used to get these parameters, and it includes the system design and the principles behind the methods used in the system. Examination of the experimental results shows that there are some problems applying the system into practical applications, it requires more accurate results. A few ideas are proposed to improve the current system.

Retrieving camera parameters is not a new issue, especially in computer vision, but for CAR applications we want minimize operations for real-time implementation pur-

poses, it should not involve calibrating the camera, or other complicated issues, like reconstructing 3D models or recovering the epipolar geometry. This is a trade-off between the simplicity, performance and accuracy.

In summary, the conclusions are:

a) The Harris corner detection is stable and consistent, but is relatively poor in localization and only gives pixel accuracy. When using a Gaussian filter to interpolate grey level values to get subpixel accuracy, special attention should be paid to the smoothness effect the filter will bring as it affects the localization of a corner;

b) Using a correlation window, the corner-based brute force search matching method is efficient in assigning 3D coordinates to some corners in the next image. The computation of the camera parameters can be done without inputting new 3D information, makes the system work as automatically as possible, but it can be computationally expensive when the number of corners becomes very large, as it uses the brute force search; it also can be unreliable when the translation, rotation and scale change ranges are large, since the number of mismatches will increase;

d) The least-squares method in computing the transformation matrix is robust and fast, but for the linear equations $Ax \approx b$ the classical least-squares method assumes the measurements of A are free of errors, all errors are confined to the observation vector b. However, this assumption is unrealistic in real cases: measuring errors, instrument errors

etc. may imply inaccuracies of matrix A as well. For a dataset with noise this method won't give the best fitting result;

e) The decomposition method used to get the camera parameters from the transformation matrix is linear, non-iterative and very fast, but "extremely susceptible to noise mainly because the orthonormal constraints associated with the rotation matrix on which this solution is based are not fully taken into account" [Phong 94];

f) This whole system gives stable result when 20 or more corners are used in the image, and in the synthetic image case, the result are very promising; for the real images, the result for the camera rotation are relatively accurate and stable, but it gives very different values for camera position, mostly because less than 10 corners are used in every image, partly because the difficulty in setting up a ideal experimental environment, which brings some immeasurable errors to the results.

7.2 Future Work

Further work could be done in order to deal with real situations:

- A better corner detector which can directly give sub-pixel position;
- A better matching method, which would give better matching results when translation, rotation and scale changes are large, and would reduce the false matches;

- A method better than least-squares to compute the transformation matrix, which tolerates more the errors in the data, and gives better fitting result. As almost all work in the area of computer vision is related in one form or another to the problem of estimating parameters from noisy data, the parameter estimation problem is usually formulated as an optimization one. Because of different optimization criteria and several possible parameterization, a given problem can be solved in many ways. The importance of choosing an appropriate method cannot be minimized. This will influence the accuracy of the estimated parameters, the efficiency of computation and the robustness to errors. When Ganapathy's method is used to compute the camera parameters, the accuracy of the transformation matrix is the key issue of the whole system;
- Finally, looking for possible methods other than Ganapathy's method which use the transformation matrix to compute the camera parameters is another choice. These methods should be less sensitive to errors in the transformation matrix. Another possibility is to improve Ganapathy's method to make it more tolerating to noise.

Bibliography

[Amanatides 92] J. Amanatides, J. Buchanan, P. poulin, A. Woo, “Optik User’s Manual Version 2.6”, Technical Report of the University of British Columbia, 1992.

[Asada 86] H. Asada, M. Brady, “The curvature primal sketch”, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.8, No.1, pp. 2-14, 1986.

[Azuma 95] R. T. Azuma, “A survey of Augmented Reality”, ACM SIGGRAPH’95 (Los Angeles, CA), August, pp. 20-1 to 20-38, 1995.

[Beaudet 78] P. R. Beaudet, “Rotational Invariant Image Operators”, Proc. of the International Conference on Pattern Recognition, pp. 578-583, 1978.

[Blaszka 94] T. Blaszka, R. Deriche, “Recovering and Charactering Image Features Using An Efficient Model Based Approach”, November Rapport de recherche N2422, INRIA, November, 1994.

[Deriche 90] R. Deriche, G. Giraudon, “Accurate Corner Detection: An Analytical Study”, Proc. 3rd International Conference on Computer Vision, pp. 66-70, 1990.

[Deriche 91] R. Deriche, G. Giraudon, “A computational Approach for Corner and Vertex Detection”, International Journal of Computer Vision, Volume 10, No.2, pp. 101-124, 1993.

[Deriche 93] R. Deriche, T. Blaszka, “Recovering and Characterizing Image Features Using An Efficient Model Based Approach”, IEEE, pp.530-535, 1993.

[Faugeras 93] O. Faugeras, B. Hotz, “Real time Correlation-based Stereo: algorithm, implementation and applications, Rapport de recherche, N2013, 1993.

[Faugeras 93-b] O. Faugeras, “Three-Dimensional Computer Vision - A Geometric Viewpoint”, Artificial Intelligence. M.I.T. Press, Cambridge, MA, 1993.

[Fournier 93] A. Fournier, A. S. Gunawan, C. Romanzin, “Common Illumination between Real and Computer Generated Scenes”, In Proceedings Graphics Interface’93, pp. 254-263, 1993.

[Freeman 77] H. Freeman, L. S. Davis, “A corner finding algorithm for chain code curves”, IEEE Trans. on Computers, No.26, pp. 297-303, 1977.

[Ganapathy 84] S. Ganapathy, “Decomposition of Transformation Matrices for Robot Vision”, Proc. Int. Conf. on Robotics and Automation, pp.130-139, 1984.

[Haralick 89] R. B. Haralick, H. Joo, C-N. Lee, X. Zhang, “Pose estimation from corresponding point data”, IEEE Transactions on Systems, Man, and Cubernetics, Vol.19, No.6, pp.1426-1445, 1989.

[Harris 88], C. Harris, “A Combined Corner and Edge Detector”, in Proc. 4th Alvey Vision Conference, pp. 147-154, 1988.

[Harris 92], C. Harris, “Geometry from Visual Motion”, Active Vision, M.I.T. Press, Cambridge, pp. 263-284, 1992.

[Horaud 90] R. Horaud, F. Veillon, T. Skordas. “Finding Geometric and Relational Structures in an Image”. Proc. First European Conference on Computer Vision, pp. 374-384, 1990.

[Huang 94] T. S. Huang, A. N. Netravali, “Motion and Structure from Feature Correspondences: A Review”, Proc. of IEEE, Vol. 82, No.2, pp. 252- 268, Feb. 1994.

[Lan 97] Z. Lan, R. Mohr, “Non-parametric Invariants and Application to Matching”, Rapport de recherche, N3246, September, 1997.

[Liu 88] Y. Liu, T. S. Huang, O. D. Faugeras, “Determination of camera location from 2D to 2D line and point correspondences”, Proc. of IEEE, 1988.

[Mohr 96] R. Mohr, B. Triggs, “Projective Geometry for Image Analysis”, A Tutorial given at ISPRS, Vienna, July, 1996. Available: http: //www.inrialpes.fr/movi.

[Phong 94] T. Q. Phong, R. Horaud, A. Yassine, P. D. Tao, “Object Pose from 2-D to 3-D Point and Line Correspondences”, Rapport de recherche N6769, INRIA, March, 1994.

[Rohr 92] K. Rohr, “Modelling and Identification of Characteristic Intensity Variations”, Image and Vision Computing, Vol10, No.2, pp.66-76, March 1992.

[Romanzin 94] C. Romanzin, “A CAR Bibliography”, unpublished report at UBC.

[Schmid 95] C. Schmid, R. Mohr, “Matching by local invariants”, Rapport de recherche, N2644, INRIA, August, 1995.

[Schmid 96] C. Schmid, R. Mohr, “Combining greyvalue with local constraints for object recognition”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 872-877, 1996.

[Semple 52] J. G. Semple, G. T. Kneebone, “Algebraic Projective Geometry”, Oxford Science Publication, 1952.

[Shapiro 95] L. S. Shapiro, “Affine analysis of image sequences”, Cambridge University Press, 1995.

[Strang 86] G. Strang, “Introduction to Applied Mathematics”, Wellesley, MS: Wellesley-Cambridge Press, 1986.

[Strat 84] T. M. Strat, “Recovering the Camera Parameters from a Transformation Matrix”, Proc. DARPA Image Understanding Workshop, Oct., pp. 264-271, 1984.

[Sutherland 74] I. E. Sutherland, “Three-dimensional data input by tablet”, Proceedings IEEE Vol. 62, No.2, pp. 453-461, 1974

[Vieville 93] T. Vieville, Q.T. Loung, “Motion of Points and Lines in the Uncalibrated Case”, Rapport de recherche, INRIA, N2054, Sept., 1993.

[Wang 92] H. Wang, M. Brady, “Corner Detection with Subpixel Accuracy”. Technical Report OUEL 1925/92, University of Oxford, 1992.

[Zhang 95] Z. Zhang, R. Deriche, O. Faugeras, Q Loung, “A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry”, Artificial Intelligence, vol.78, no.1-2, pp. 87-119, 1995.