CAMERA POSE ESTIMATION FROM A STEREO SETUP

Sébastien Gilbert

A Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Applied Science, Electrical Engineering

February 2005

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Contents

List of Tables

List of Figures

Abstract

This thesis addresses the problem of estimation of the camera poses with respect to a rigid object, which is equivalent to the problem of tridimensional registration of a moving rigid object before fixed cameras. Matching, tracking and 3D reconstruction of feature points by a stereoscopic vision setup allows the computation of the homogeneous transformation matrix linking two consecutive scene captures. Robustness to errors is provided by the scene rigidity constraint. Accumulation of error is compensated through loop detection in the calculated camera poses. Experimental results show the validity of the obtained camera poses.

Acknowledgements

I would like to thank my supervisors, Dr Robert Laganière and Dr Gerhard Roth, for their constant support and countless advices. Dr Étienne Vincent and Jia Li wrote the image acquisition software I used.

I am especially grateful for my spouse Sophie Boucher and my children Myriam and Éloi who have gone through two years of sacrifices, allowing me to complete this program.

Finally, this work would not have been possible without the financial support of the Natural Sciences and Engineering Research Council of Canada, through a Graduate Studies Scholarship.

Chapter 1

Introduction

Computer vision is a young discipline. Its implementations used to be regarded as costly, complex solutions to simple problems. After all, the human visual system and the human brain are tremendous scene understanding machines! It is amazing how easily a human being can infer 3D information from a single, noisy image. Anyone who has experience in programming vision-related software has been stunned by this simple fact: it is so easy to perform with your own eyes and so hard to tell a computer to do it. Nevertheless, hard work is rewarded, for there are countless situations for which a computer vision algorithm will perform better than a human operator.

Trucco and Verri [1] provide the following definition of the scope of computer vision:

> "A set of computational techniques aimed at estimating or making explicit the geometric and dynamic properties of the 3-D world from digital images"

The roots of machine "understanding" of images can be found in the sixties, as the United States Postal Service implemented an optical character recognition system to sort mail. As its name suggests, the first recognition devices were optical. With the advent of video cameras, frame grabbers and digital cameras, computers had an increasing role in vision tasks. Since then, computer vision has become a subject

of intensive research around the world. Its commercial applications are increasing, as the computers power keep up with Moore's law. Even more critical, the price of digital cameras has been going down substantially in the last few years. This makes good quality digital images ubiquitous.

Numerous research areas can be identified within computer vision science. Pattern recognition is a very important one, and perhaps the most relevant to artificial intelligence. The goal of pattern recognition is to identify an object from one or many images of it. A pattern recognition algorithm will be evaluated according to its ability to produce large rates of correct identification under noisy conditions, and under a wide range of perceptual distortions (change in scale, orientation or illumination conditions). Typically, a trade-off will be reached between the discriminating power of an algorithm and its robustness to distortions.

Nowadays, there is an ever-increasing demand for surveillance and security applications of computer vision. Biometrics, for instance, aims at identifying individuals from images of their face, fingertips or iris. Face identification is an example of an application that could benefit from a combined use of pattern recognition and stereo vision.

The word *stereo* comes from the Greek *stereos*, which means solid. Stereo vision is the acquisition and analysis of 3D information, through the capture of scene images from different points of view. It can be performed by using more than one camera, or one moving camera with respect to a rigid scene. Once again, the human visual system itself is an amazingly powerful stereo vision system, but it lacks quantitative accuracy in the evaluation of distances. As will be demonstrated in the course of this thesis, a calibrated stereo system can easily achieve an accuracy of distance measurement that most human beings cannot approach. This makes stereo vision setups ideally suited for non-contact measurements. Parts inspection systems, space robotics and crime scene analysis software exploit this interesting potential.

1.1 Problem Description

One of the main challenges inherent to the use of images from a large number of viewpoints is the issue of camera pose estimation. For 3D reconstruction to be possible, the location and orientation of the cameras at the instant of capture must be accurately known. This is typically achieved through *bundle adjustment*, in which, assuming the intrinsic calibration parameters of the cameras and a large number of matches between pairs of views are known, the 3D location of the feature points and the position of the cameras are computed. This method relies on a human operator, who has to supply the matches since there is typically a small number of widely separated views.

Bundle adjustment is widely accepted, and commercial software is available. It is used for a wide spectrum of applications, such as accident reconstruction, animation and graphics, archaeology, forensics, engineering and architecture [1]. The main drawback of bundle adjustment is its instability. In many situations, the algorithm will fail to converge to an accurate solution. To overcome this problem, it is recommended that the user starts with a small subset of the available pictures, with a small number of feature points that can be seen in many pictures. Once the algorithm succeeded in converging to a first reasonable solution, additional intermediate pictures and more feature points can be added to improve the accuracy. In some instances, even with a few pictures and a small number of feature points, bundle adjustment fails to converge, forcing the user to eliminate some apparently good feature points. This lack of robustness can be related to the iterative nature of bundle adjustment, which implies initial estimates of the camera positions. If these estimates are very far from the actual solution, the algorithm may fail to converge.

This problem is amplified when one wants to automate the whole process. Matches between narrowly separated views can be found automatically through correlation. Unfortunately, nothing can guarantee the matches will all be good. Bad matches will definitely prevent the convergence of the bundle adjustment algorithm. The

[1] www.photomodeler.com

constraint on the maximum separation between the views prevents the use of a small number of widely separated pictures, as a first step. Therefore, in a typical scenario, a large number of narrowly separated pictures should be used from the start, with the well-known instability that such a situation implies. On top of that, a small percentage of bad matches will make proper convergence very unlikely. Automation is where the proposed method becomes interesting.

It is the goal of this thesis **to provide a technique that fully automates the process of estimating the camera poses, with respect to a rigid scene.** It will be assumed that a stereo setup is available and the position change from one capture to the next is small, to simplify feature points tracking. The cameras will move rigidly with respect to each other, preserving the stereo configuration. The system will have to be robust with respect to matching and tracking errors. Additionally, it will have to compensate for error accumulation, as a post-processing step.

Scene feature points will be detected and matched, so their 3D locations will be computed by the calibrated stereo setup. Then, they will be tracked in each camera view, and reconstructed again after the rigid motion. The two clouds of 3D points will be robustly registered, allowing the calculation of the new camera positions. This process will be repeated along a sequence. The error accumulation in the camera positions will have to be corrected through the detection of loops in the general path of the cameras.

1.2 Contributions

This thesis brings contributions to the specific fields of camera calibration and 3D reconstruction in the context of a stereoscopic system.

In Chapter 2, we experimentally found the optimal configuration of the stereo setup we were using by finding the baseline distance between the cameras such that the reconstruction error doesn't improve anymore with an increase of baseline.

We also experimentally showed, in Section 3.3, that the calibration of the stereo

setup can be done by independently calibrating the cameras individually. The stereo geometry, as encoded in the fundamental matrix, can be constructed from these calibration data, and there is no need to use the eight-point algorithm.

To solve the stereo pair positioning problem, we implemented an algorithm to register two clouds of 3D points in which we integrated a random sampling strategy to make it robust to the unavoidable presence of outliers.

The problem of camera motion tracking with a stereo setup has been studied by several researchers ([22], [23], [24]). They however limited their analysis to a single registration or a short sequence, such that the cameras field of view would not change drastically from the first to the last picture. To the best of our knowledge, long stereo sequences, where registration error accumulation is significant, have never been reported. A fortiori, no error correction scheme based on automatic loop detection has been reported.

We proposed a novel automatic loop detection technique. When, in the course of the acquisition process, a camera finds itself in a position close by a previously visited location, the system automatically rectifies the image to facilitate matching with the past images. Such detection of path intersection has the benefit of short-circuiting the undesirable accumulation of errors in pose estimation.

In addition, to reduce the impact of error accumulation, we proposed a novel error correction scheme based on the computation of a correction matrix that can be interpolated to the intermediate views. We showed experimentally that the proposed error correction scheme reduces the accumulated error, through a comparison with a commercial bundle adjustment software.

Finally, in order to demonstrate the validity of our approach, we have proposed two applications: object reconstruction and scene augmentation. We have therefore implemented a shape-from-silhouette algorithm that uses the camera positions computed by our approach. It results an original free-hand 3D object reconstruction system. The fact that, from our approach, the position of the object with respect to the camera system is known allows one to add a virtual reference frame to the scene

that rigidly moves with the observed object.

1.3 Thesis organization

Chapter 2 and the appendices provide an overview of the theoretical and mathematical tools that will be used to achieve the proposed goal. Chapters 3 through 6 provide detailed experimental results on various aspects of the project. Of particular importance is chapter 4, which provides an overview of the whole proposed method.

Chapter 2

The Stereoscopic Setup

An image is a projection of a tridimensional scene on a plane. The distance from a given point to the projection center is lost in the process. To recover this information, multiple projections of the same point must be performed. For instance, evolved animals have two eyes in order to get a mental volumetric representation of the world. In robotics applications, arrays of cameras are often distributed around the working area. Once the geometry of the setup is known along with a few key parameters of the sensors, tridimensional euclidian coordinates of feature points can be computed. This process is referred to as *3D reconstruction*.

A stereoscopic setup is a set of at least two cameras which record the same scene from different viewpoints. The redundancy of information increases with the number of cameras, yielding to a more and more precise reconstruction. In this chapter, we will present the theoretical bases of a two-cameras stereo setup. The calibration procedures will be covered in Chapter 3.

2.1 Linear Pinhole Camera Model

In the framework of the present thesis, we will be using a linear pinhole camera model. A pinhole camera is one that performs imaging through a pinhole aperture. In reality, cameras use a large aperture and image the input scene with a lens or a combination

of lenses. The reason for doing so is one of efficiency. The more light goes through the aperture, the easier it is for detectors to work. A pinhole camera would be simpler to manufacture, but would have to work with very small light intensities. The optics of a real camera aim at imaging the input scene on the image plane, just what a pinhole camera does (actually, a pinhole camera images on any plane behind the pinhole: it is always in focus). For these reasons, the pinhole camera model is suitable for most real cameras.

The main problems introduced by optics are the blurring effect caused by an object being far from the object plane and the radial distortions created by lenses with a large numerical aperture. We will assume that the scene being looked at is within the depth of field of the camera, and therefore no feature points will be blurred by an out-of-focus effect. Regarding the radial distortions, we will neglect it in the first approximation, yielding to a *linear model*. Radial distortions might originate from difficulties in manufacturing the exact optimal shape of a lens, or from a trade-off in the design itself. In some cases, some distortions at the border of the image might be a reasonable price to pay to get a wider field of view or cheaper manufacturing cost. The webcam industry is a good example of such a marketing choice. These devices produce strongly distorted, wide field of view images, at a very low price. We will be using narrow field of view cameras (f = 0.12 m) and assume the radial distortions will be small. If it happens that, for a given set of cameras, it is not the case, radial distortion coefficients have to be computed. Along with the location of the optical center of the camera, they allow one to translate distorted pixel values into undistorted pixel values, for which the linear pinhole camera model holds.

The pinhole camera model images the scene by allowing the light to go through a small aperture (the pinhole). The inverted image can be seen as a projection on the image plane located at an arbitrary distance behind the pinhole. This distance is called the focal length (although there is no lens involved in a pinhole camera model). An image projected on a plane with inverted axes behind the pinhole is mathematically equivalent to a non-inverted image formed at a focal distance in front

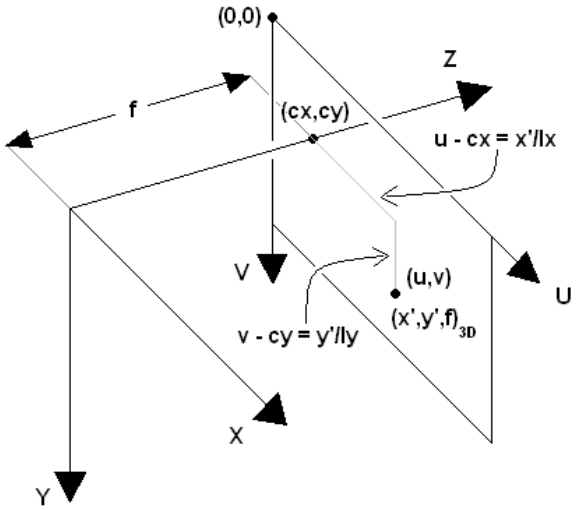


Figure 2.1: Linear pinhole camera model

of the pinhole. For simplicity, we will use the later interpretation, as depicted in Figure 2.1. This mathematical feature does not have any physical meaning: it is just a way to avoid the double inversion of the pixel coordinates.

Let us assume an object point is being imaged on the image plane of a camera. The 3D coordinates of this image point are known, and we want to compute the location of this point in pixels. This task is carried out by the *intrinsic calibration matrix*. It is a 3×3 matrix that converts 3D coordinates in the image plane into homogeneous pixel coordinates.

In Figure 2.1, a 3D point (X', Y', f) located in the image plane of the camera is shown, along with its relationship with the pixel values. The *principal point*, (c_x, c_y) is the point, expressed in pixels, where the optical axis (the Z axis) crosses the image plane. The horizontal and vertical spacing of the sensor elements in the sensor array are denoted by l_x and l_y respectively. Along with the focal distance, f, they represent the *intrinsic calibration parameters* of the camera. The equation linking the pixel values (u, v) with the 3D coordinates (X', Y', f) and the intrinsic calibration parameters is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{l_x} & 0 & \frac{c_x}{f} \\ 0 & \frac{1}{l_y} & \frac{c_y}{f} \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ f \end{bmatrix} \tag{2.1}$$

Equation (2.1) gives us an insight of the way an arbitrary 3D point, expressed in the reference system of the camera, will be projected in the image plane. Any point $\vec{x} = (X, Y, Z)$ can be seen as the product of a scaling factor α times the 3D coordinates of a 3D point in the image plane $\vec{x'} = (X', Y', f)$. All points belonging to the line $\vec{l} = \alpha[X', Y', f]^T$ will be projected in the image plane at position (X', Y', f) since the origin is the center of projection. Hence, for any point $\vec{x} = (X, Y, Z)$ as expressed in the reference system of the camera, its pixel coordinates are given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \frac{1}{l_x} & 0 & \frac{c_x}{f} \\ 0 & \frac{1}{l_y} & \frac{c_y}{f} \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

The parameter $\frac{1}{\alpha}$ must be adjusted such that the third element in the left-hand side of equation (2.2) is unity. This procedure reflects the loss of information that takes place in imaging: an infinity of 3D points are imaged on a single 2D point. Since this parameter is floating and has no physical significance, it can absorb the $\frac{1}{f}$ factor of equation (2.2) to yield:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} \frac{f}{l_x} & 0 & c_x \\ 0 & \frac{f}{l_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.3}$$

$$\vec{u} = \lambda K \vec{x} \tag{2.4}$$

where K is the intrinsic calibration matrix. K is a property of a camera. In general, the focal length f of a camera is supplied by the manufacturer, but the focal length - to - pixel dimensions ratios, $\frac{f}{l_x}$ and $\frac{f}{l_y}$, along with the principal point (c_x, c_y) must be computed through a calibration procedure.

2.2 Projection Matrix

Equation (2.4) allows one to compute the pixel values of the image of a 3D point, assuming this point is expressed in the reference system of the camera. A more general approach would require the computation of the pixel values of an image, starting from a 3D point expressed in an arbitrary reference system. This work is carried out by the *projection matrix*, P. It is a 3×4 matrix that premultiplies a 4×1 column vector $\vec{X}$, that expresses homogeneous 3D coordinates in a global reference system, to yield pixel values.

The construction of the projection matrix P can be done in three steps:

1. Convert the coordinates in the global reference frame into coordinates in the camera reference frame
2. Suppress the "1" at the bottom of the homogeneous 3D coordinates
3. Project the 3D coordinates in the reference system of the camera in the image plane through the intrinsic calibration matrix K

Conversion of Coordinates in the Global Reference Frame into Coordinates in the Camera Reference Frame

As covered in Section A.2, this task is carried out by a homogeneous transformation matrix. According to the convention expressed in (A.46), the homogeneous transformation matrix linking the two coordinates systems will be defined by:

$$\vec{X}|_{global} = Q_{cam/world}\vec{X}|_{cam} \tag{2.5}$$

Since we are interested in the coordinates expressed in the reference frame of the camera, given some coordinates expressed in the global reference frame:

$$\vec{X}|_{cam} = Q^{-1}_{cam/world}\vec{X}|_{global} \tag{2.6}$$

Suppression of the "1" at the Bottom of the Homogeneous 3D Coordinates

In order to have a 3×1 column vector $\vec{x}$ to apply to the intrinsic calibration matrix, the fourth element of the homogeneous coordinates $\vec{X}$ must be discarded. This is done by premultiplying the homogeneous coordinates by the $[I|0]_{3\times4}$ matrix:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{camera} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{camera} \tag{2.7}$$

$$\vec{x} = [I|0]_{3\times4}\vec{X} \tag{2.8}$$

Projection of the 3D Coordinates in the Reference System of the Camera in the Image Plane

As seen in equation (2.4), the projection of a non homogeneous 3D point $\vec{x}$ as expressed in the reference system of the camera can be obtained by using the intrinsic calibration matrix K.

Combining (2.4), (2.8) and (2.6) yields to:

$$\vec{u} = \lambda K\vec{x} = \lambda K \times [I|0]_{3\times4}\vec{X}|_{camera} = \lambda K[I|0]_{3\times4}Q^{-1}_{cam/world}\vec{X}|_{world} \tag{2.9}$$

$$\vec{u} = \lambda P\vec{X}|_{world} \tag{2.10}$$

where

$$P = K[I|0]_{3\times4}Q^{-1}_{cam/world} \tag{2.11}$$

The projection matrix P is the most compact representation of how a camera captures the information of a scene. The projection matrix encapsulates the intrinsic calibration parameters of the camera, along with the camera's location and orientation with respect to a global reference frame. Therefore, performing the complete calibration of a camera is synonymous to computing its projection matrix.

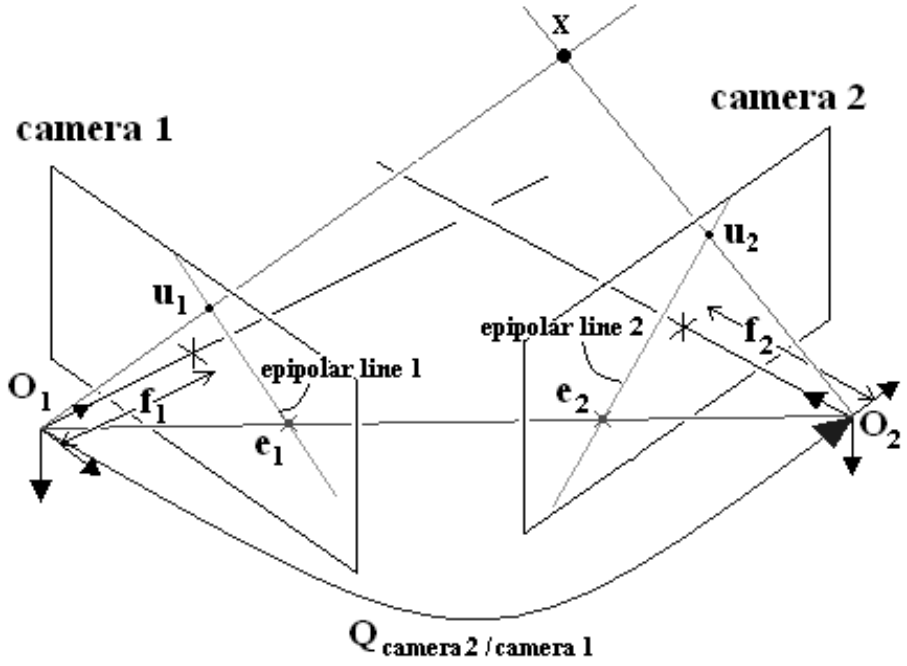


Figure 2.2: Geometry of a two cameras stereo setup

2.3 Stereoscopic Setup Geometry

The geometry of a general stereoscopic setup is depicted in Figure 2.2. It is worth noticing that we won't assume at this point that the camera frames are parallel. In chapter 4, when we will be concerned with matching, we will add the requirements that the two cameras have their Y-axes (or X-axes) nearly parallel and their separation is not too big. These requirements come from the matching algorithm that we will implement, based on window correlation, but they are not features of the general stereoscopic geometry we are discussing now.

A stereo setup is essentially defined by the homogeneous transformation matrix relating the reference systems of the two cameras, $Q_{cam2/cam1}$, and the focal lengths of the cameras, f_1 and f_2. The most important feature of this configuration is the segment $\overline{O_1O_2}$ that joins the two centers of projection. The points where this segment intersects the image planes of the cameras are called the *epipoles*, $\vec{e}_1$ and $\vec{e}_2$.

From the geometry of the configuration, an interesting observation can be formulated. Let us assume the image point $\vec{u}_1$ of the scene point $\vec{X}$ has been identified in image 1. It is known that the 3D point $\vec{X}$ must lie in the line defined by the two points $\vec{O}_1$ and $\vec{x}'_1$ (the 3D point in the image plane corresponding to the pixels point $\vec{u}_1$). This line, when projected in the image plane of camera 2, yields to an *epipolar line*. This name comes from the fact that all the epipolar lines must go through the epipole of a camera. Using this information, finding the image point $\vec{u}_2$ of the scene point $\vec{X}$ can be restricted to searching along the epipolar line corresponding to $\vec{u}_1$. Identifying matches is usually done through correlation. Without this knowledge, finding the match for $\vec{u}_1$ requires a 2D search over the entire image. Knowing the location of the epipoles allow us to turn this task into a 1D search of correlation matches.

2.4 Essential Matrix and Fundamental Matrix

The essential and fundamental matrices express mathematically what was observed in section 2.3, namely the image point - to - epipolar line correspondence of a stereo setup. The essential matrix E is a 3×3 matrix that allows to compute a 3D line in the image plane of camera 2, given a 3D point $\vec{x'}_1$ in the image plane of camera 1, and vice-versa:

$$\vec{x'}_2^T E \vec{x'}_1 = 0 \tag{2.12}$$

The fundamental matrix is a 3×3 matrix that expresses the same relationship, with pixel points. Given an image point $\vec{u}_1$ from camera 1, the fundamental matrix F allows to compute the corresponding epipolar line in image 2, and vice-versa:

$$\vec{u}_2^T F \vec{u}_1 = 0 \tag{2.13}$$

The explicit form of the essential and the fundamental matrices are the following:

$$E = R^T_{cam2/cam1} S_{cam2/cam1} \tag{2.14}$$

$$F = K_2^{-T} R^T_{cam2/cam1} S_{cam2/cam1} K_1^{-1} = K_2^{-T} E K_1^{-1} \tag{2.15}$$

where

$$S_{cam2/cam1} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}_{cam2/cam1} \tag{2.16}$$

T_x, T_y, T_z being the components of the translation vector $\vec{T}$ and K_1 and K_2 being the intrinsic calibration matrices.

Proof. **Preliminary result:**

$$\vec{V} \times \vec{T} = S^T \vec{V} \tag{2.17}$$

where

$$S \equiv \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.18}$$

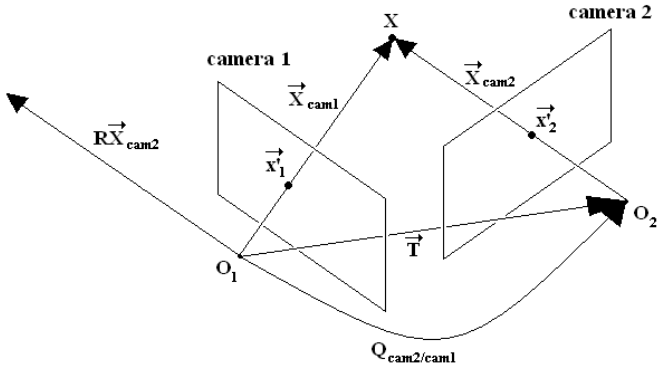


Figure 2.3: Coplanarity of $R_{cam2/cam1}\vec{X}|_{cam2}$, $\vec{X}|_{cam1}$ and $\vec{X}|_{cam2/cam1}$

Proof of 2.17

$$\vec{V} \times \vec{T} = \det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ V_x & V_y & V_z \\ T_x & T_y & T_z \end{bmatrix} = \begin{bmatrix} V_y T_z - V_z T_y \\ V_z T_x - V_x T_z \\ V_x T_y - V_y T_x \end{bmatrix}$$

$$= \begin{bmatrix} 0 & T_z & -T_y \\ -T_z & 0 & T_x \\ T_y & -T_x & 0 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = S^T \vec{V} \tag{2.19}$$

Let us define a two cameras stereo setup. A 3D point $\vec{X}$ is imaged in both cameras, corresponding to the 3D points $\vec{x}'_1$ and $\vec{x}'_2$ respectively. The reference systems of the two cameras are related by a rotation $R_{cam2/cam1}$ and a translation $\vec{T}_{cam2/cam1}$, such that:

$$\vec{X}|_{cam1} = R_{cam2/cam1}\vec{X}|_{cam2} + \vec{T}_{cam2/cam1} \tag{2.20}$$

In order for (2.20) to be possible, the vector $R_{cam2/cam1}\vec{X}|_{cam2}$ must lie in the plane defined by the vectors $\vec{X}|_{cam1}$ and $\vec{T}_{cam2/cam1}$ (see Figure 2.3). Therefore, their mixed double product is zero:

$$(R_{cam2/cam1}\vec{X}|_{cam2} \times \vec{T}_{cam2/cam1})^T \cdot \vec{X}|_{cam1} = 0 \tag{2.21}$$

Inserting (2.17) into (2.21):

$$(S^T_{cam2/cam1} R_{cam2/cam1} \vec{X}|_{cam2})^T \cdot \vec{X}|_{cam1} = 0 \tag{2.22}$$

$$\vec{X}|^T_{cam2} R^T_{cam2/cam1} S \vec{X}|_{cam1} = 0 \tag{2.23}$$

Since $\vec{x'}_1 = \alpha \vec{X}|_{cam1}$ and $\vec{x'}_2 = \beta \vec{X}|_{cam2}$ where α and β are scalars, (2.23) becomes:

$$\frac{1}{\alpha\beta} \vec{x'}^T_2 R^T_{cam2/cam1} S_{cam2/cam1} \vec{x'}_1 = 0 \tag{2.24}$$

$$\vec{x'}^T_2 R^T_{cam2/cam1} S_{cam2/cam1} \vec{x'}_1 = 0 \tag{2.25}$$

$$\vec{x'}^T_2 E \vec{x'}_1 = 0 \tag{2.26}$$

where

$$E = R^T_{cam2/cam1} S_{cam2/cam1} \tag{2.27}$$

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.28}$$

Inserting (2.4) into (2.25):

$$(\frac{1}{\lambda_2} K_2^{-1} \vec{u}_2)^T R^T_{cam2/cam1} S_{cam2/cam1} (\frac{1}{\lambda_1} K_1^{-1} \vec{u}_1) = 0 \tag{2.29}$$

$$\vec{u}^T_2 K_2^{-T} R^T_{cam2/cam1} S_{cam2/cam1} K_1^{-1} \vec{u}_1 = 0 \tag{2.30}$$

$$\vec{u}^T_2 F \vec{u}_1 = 0 \tag{2.31}$$

where

$$F = K_2^{-T} R^T_{cam2/cam1} S_{cam2/cam1} K_1^{-1} \tag{2.32}$$

□

2.4.1 Eigenvalues and Rank of E

Computing the eigenvalues of E yields to the following characteristic equation:

$$\lambda(\lambda^2 + \lambda b(R,\vec{T}) + c(R,\vec{T})) = 0 \tag{2.33}$$

where

$$b(R,\vec{T}) \equiv (r_{12} - r_{21})T_x + (r_{20} - r_{02})T_y + (r_{01} - r_{10})T_z \tag{2.34}$$

$$\begin{aligned} c(R,\vec{T}) \equiv (r_{11}r_{22} - r_{12}r_{21})T_x^2 + (r_{00}r_{22} - r_{02}r_{20})T_y^2 \\ +(r_{00}r_{11} - r_{01}r_{10})T_z^2 + (r_{02}r_{21} - r_{01}r_{22} + r_{12}r_{20} - r_{10}r_{22})T_xT_y \\ +(r_{01}r_{12} - r_{02}r_{11} + r_{10}r_{21} - r_{11}r_{20})T_xT_z \\ +(r_{01}r_{20} - r_{00}r_{21} + r_{02}r_{10} - r_{00}r_{12})T_yT_z \end{aligned} \tag{2.35}$$

The eigenvalues of E are therefore

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}_E = \begin{bmatrix} 0 \\ \frac{-b(R,\vec{T}) + \sqrt{b^2(R,\vec{T}) - 4c(R,\vec{T})}}{2} \\ \frac{-b(R,\vec{T}) - \sqrt{b^2(R,\vec{T}) - 4c(R,\vec{T})}}{2} \end{bmatrix} \tag{2.36}$$

Since E has rank 2 and the intrinsic calibration matrices are full-rank (see(2.2)), F has rank 2. Hence, the essential and the fundamental matrices are defined up to a scale factor. In order to facilitate comparisons of different matrices, it is common practice to set a scaling factor such that one of the non-zero elements of the matrix (typically the upper-left entry) is unity.

2.4.2 Eight-Point Algorithm

The eight-point algorithm allows one to compute the fundamental matrix of a stereo setup from a set of matches. It was first introduced by Longuet-Higgins [2]. Each of the matches $(\vec{u}_1, \vec{u}_2)$ can be used to build a homogeneous linear equation, from (2.13):

$$\vec{u}_2^T F \vec{u}_1 = 0 \tag{2.37}$$

The problem of computing the entries of F can be seen as searching for nine numbers, from the homogeneous linear equations in nine unknowns provided by each match. It can be solved by computing the non-trivial solution of a minimum set of eight homogeneous linear equations. As a consequence, the fundamental matrix can be computed from a minimum of eight matches.

The singularity of the fundamental matrix must be enforced [1]. This can be done by performing singular value decomposition (SVD) on the obtained matrix F': $F' = UD'V^T$. If the decomposed matrix is singular, the diagonal matrix D' will contain a null value. In practice, the smallest singular value may not be identically zero. To obtain a genuine singular matrix, the diagonal matrix D' must be replaced by another diagonal matrix D whose entries are equal to those of D', except for the smallest entry of D' that must be replaced by a 0. Then, $F = UDV^T$ is the singular matrix that is the closest of F', in the sense of the Frobenius norm [3].

This simple version of the eight-point algorithm presents a flaw. It is well known that its implementation yields to numerical instabilities. Hartley [3] showed the subtle origin of this problem. While evaluating the least-square solution of the system $A\vec{f} = 0$ where $\vec{f}$ contains the entries of F, one has to compute the least eigenvector of A^TA. This square matrix typically has a wide range of variation in the orders of magnitude of its entries, when usual pixel values are used, with the origin as the upper left corner. Hartley showed that, by proper scaling and translation of the pixel values, such that their transformed centroid is (0, 0, 1) and their average modulus is around 1, the entries of the obtained A'^TA' are much more uniform. This procedure is referred to as the normalization of image points. It yields to a more stable solution of $A'f' = 0$. The original fundamental matrix can then be retrieved by undoing the transformations, i.e. $F = T_2^T F' T_1$, where T_1 and T_2 are the transformation matrices applied to the original image coordinates of images 1 and 2 respectively.

The eight-point algorithm allows one to estimate the fundamental matrix from matches, with no information about the intrinsic and extrinsic calibration of the cameras. It is therefore a powerful tool in the case of uncalibrated images. In the

case where the calibration of the setup is known, the next chapter will address the issue of whether it is worthwhile to compute the fundamental matrix through the eight-point algorithm.

2.5 3D Reconstruction

Tridimensional reconstruction is the task of computing the euclidian coordinates of a feature point from its pixel coordinates in multiple images and the knowledge of the calibration parameters of the stereo setup. Two alternative approaches will be presented in this section: the triangulation and the least-square solving from the projection matrices. These two techniques aim at minimizing different error quantities. The triangulation, which is presented for two cameras, finds the 3D point that minimizes its *3D distance* with two non-crossing lines in space. In order words, it returns the middle of the segment perpendicular to both lines. Least-square solving from the projection matrices aims at minimizing an algebraic quantity such that $\vec{u} = \lambda P\vec{X}$ holds well for every camera. Both methods are valid and produce very similar results in practice.

2.5.1 Triangulation

This method is described in [1].

Figure 2.4 shows the geometry of two cameras projecting the images $\vec{u}_1$ and $\vec{u}_2$ of the 3D point $\mathbf{x}$. Let x'_1 and x'_2 be the 3D points in the image planes of camera 1 and 2 respectively, located at the image points $\vec{u}_1$ and $\vec{u}_2$. In an ideal situation, the extension of the lines $\overrightarrow{O_1x'_1}$ and $\overrightarrow{O_2x'_2}$ should cross each other in space at the location of the projected 3D point $\mathbf{x}$. In reality, the two lines may not cross. We will be searching for the point $\mathbf{x}$ that is the middle of the segment $\overrightarrow{x_1x_2}$, perpendicular to

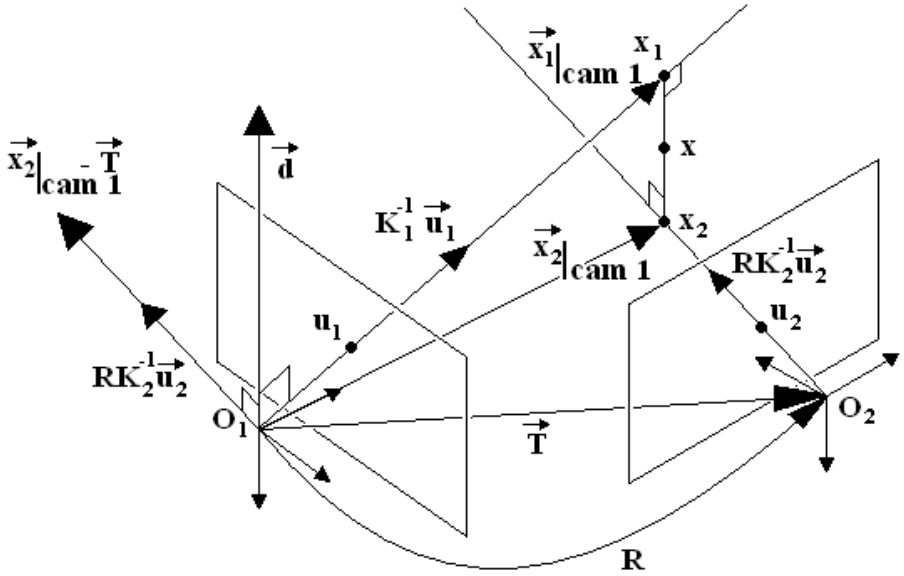


Figure 2.4: Geometry of the triangulation procedure

both lines. From (2.4),

$$\vec{u}_j = \lambda_j K_j \vec{x}_j|_{camj} \tag{2.38}$$

$$\vec{x}_1|_{cam1} = \frac{1}{\lambda_1} K_1^{-1} \vec{u}_1 \tag{2.39}$$

$$\vec{x}_2|_{cam2} = \frac{1}{\lambda_2} K_2^{-1} \vec{u}_2 \tag{2.40}$$

Applying the convention of Section A.2:

$$\vec{x}_2|_{cam1} = R\vec{x}_2|_{cam2} + \vec{T} \tag{2.41}$$

$$\vec{x}_2|_{cam1} = \frac{1}{\lambda_2} R K_2^{-1} \vec{u}_2 + \vec{T} \tag{2.42}$$

Let us define the vector $\vec{d}$ (see Figure 2.4), that is proportional to the cross product of $\vec{x}_1|_{cam1}$ and $(\vec{x}_2|_{cam1} - \vec{T})$:

$$\begin{aligned} \vec{d} &\equiv \lambda_1 \lambda_2 \vec{x}_1|_{cam1} \times (\vec{x}_2|_{cam1} - \vec{T}) \\ &= K_1^{-1} \vec{u}_1 \times R K_2^{-1} \vec{u}_2 \end{aligned} \tag{2.43}$$

The vector $\vec{d}$ is therefore parallel to the vector $\overrightarrow{x_1x_2}$. Let us now define three scalars a, b and c such that the path $O_1x_1x_2O_2O_1$ forms a closed loop:

$$aK_1^{-1}\vec{u}_1 + b\vec{d} + cRK_2^{-1}\vec{u}_2 - \vec{T} = 0 \quad (2.44)$$

$$aK_1^{-1}\vec{u}_1 + b[K_1^{-1}\vec{u}_1 \times RK_2^{-1}\vec{u}_2] + cRK_2^{-1}\vec{u}_2 = \vec{T} \quad (2.45)$$

Equation (2.45) provides three linear equations in three unknowns, a, b and c. Once this system is solved for a given match $(\vec{u}_1, \vec{u}_2)$, the location of the point $\mathbf{x}$ can be calculated:

$$\vec{x}|_{cam1} = aK_1^{-1}\vec{u}_1 + \frac{1}{2}b[K_1^{-1}\vec{u}_1 \times RK_2^{-1}\vec{u}_2] \quad (2.46)$$

2.5.2 Least-Squares Solving from the Projection Matrices

An alternative approach for 3D reconstruction under the assumption of complete knowledge of the calibration involves the use of the projective equation (2.10). Let us assume we have n cameras imaging a point $\mathbf{x}$ with global homogeneous coordinates $\vec{X}$, through their respective projection matrices P_i. Each of the cameras provides us with three non-linear equations in four unknowns, λ_i, X, Y, and Z.

$$\lambda_i \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix}_i \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_i \quad (2.47)$$

As discussed in Section 2.1, the λ_i parameter reflects the loss of information that occurs when a 3D point in projected into a 2D image plane, and must be set such that the homogeneity of the equation is preserved (the third entry of $\vec{u}_i$ is one). This information is not relevant, since we are searching for the 3D coordinates of $\mathbf{x}$, (X, Y, Z). As a consequence, eliminating the λ_i from the set of unknowns would greatly simplify the formulation. This can be performed by a few algebraic manipulations, which also make the system to be solved linear.

Let us isolate λ_i:

$$\lambda_i = \frac{1}{(p_{20}X + p_{21}Y + p_{22}Z + p_{23})_i} \tag{2.48}$$

Inserting (2.48) into the two first lines of (2.47):

$$(p_{00} - up_{20})_i X + (p_{01} - up_{21})_i Y + (p_{02} - up_{22})_i Z = (up_{23} - p_{03})_i \tag{2.49}$$

$$(p_{10} - vp_{20})_i X + (p_{11} - vp_{21})_i Y + (p_{12} - vp_{22})_i Z = (vp_{23} - p_{13})_i \tag{2.50}$$

Each camera providing 2 linear equations in three unknowns (X, Y, Z), we get a system of $2n$ linear equations in three unknowns:

$$\begin{bmatrix} (p_{00} - up_{20})_1 & (p_{01} - up_{21})_1 & (p_{02} - up_{22})_1 \\ (p_{10} - vp_{20})_1 & (p_{11} - vp_{21})_1 & (p_{12} - vp_{22})_1 \\ (p_{00} - up_{20})_2 & (p_{01} - up_{21})_2 & (p_{02} - up_{22})_2 \\ (p_{10} - vp_{20})_2 & (p_{11} - vp_{21})_2 & (p_{12} - vp_{22})_2 \\ \dots & \dots & \dots \\ (p_{00} - up_{20})_n & (p_{01} - up_{21})_n & (p_{02} - up_{22})_n \\ (p_{10} - vp_{20})_n & (p_{11} - vp_{21})_n & (p_{12} - vp_{22})_n \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (up_{23} - p_{03})_1 \\ (vp_{23} - p_{13})_1 \\ (up_{23} - p_{03})_2 \\ (vp_{23} - p_{13})_2 \\ \dots \\ (up_{23} - p_{03})_n \\ (vp_{23} - p_{13})_n \end{bmatrix} \tag{2.51}$$

The system (2.51) can be solved using a least-squares linear method.

2.6 Configuration of the Cameras

The spatial configuration of a pair of cameras (the relative orientation and location) have an impact on the quality of the 3D reconstruction. From inspection of Figure 2.4, one can infer that the closer the two cameras are to each other, the more parallel the two projection rays will be. Hence, the more imprecise the intersection point will be, especially in the Z-direction. As can be observed from Figure 2.5, if one wants to increase the *baseline* (the distance between the centers of projection of the cameras), one must increase the angle between the Z-axes of the cameras in order to keep a given working area.

A limiting factor on the angle between the Z-axes of the cameras is the ability to perform point matching between the views. As will be discussed in Section 4.1, we will

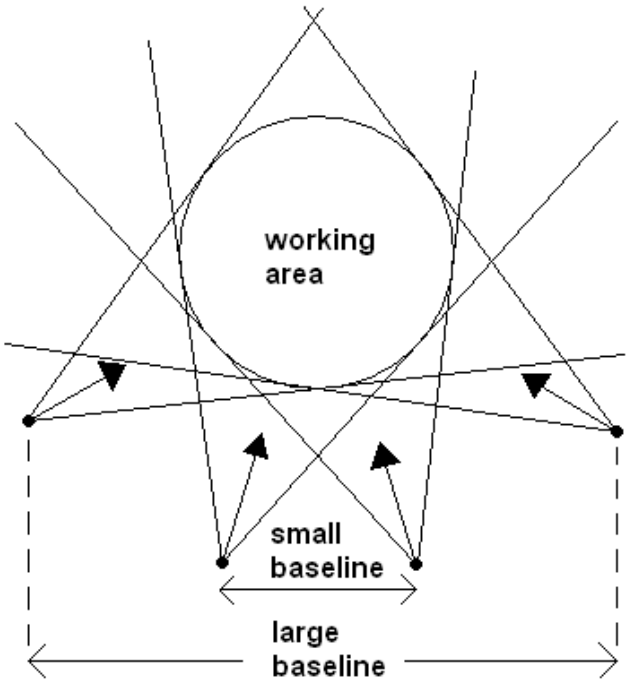


Figure 2.5: Necessity to increase the angle between the cameras Z-axes, as the baseline is increased

use correlation to identify matches. This family of techniques cannot tolerate large perspective distortion in the surrounding of a pixel of interest. As far as matching is concerned, the more parallel the camera frames, the better.

We performed an experiment in which we built three stereo setups with different baselines (0.139 m, 0.416 m and 0.756 m). The angles between the Z-axes of the two cameras were adjusted in such a way that a given working volume was preserved, resulting in angles of 0.112 rad, 0.463 rad and 1.05 rad respectively. Reconstruction was performed using least-squares solving from the projection matrices, as described in Section 2.5.2. A calibration pattern (Figure 2.6) was used, allowing easy detection of its feature points with sub-pixel resolution, through Hough transform. The position of the calibration pattern with respect to the table was measured with a ruler. This procedure provides the ground truth value of the feature points position, with an estimated accuracy of a fraction of a millimeter.

Figure 2.7 shows the reconstruction error ($|\vec{x}_{calculated} - \vec{x}_{measured}|$) averaged over the 20 feature points of a calibration pattern as a function of the Z position of the calibration pattern, for three different baselines. It can be observed that the reconstruction error is higher for the stereo setup with the smallest baseline, as expected. No significant difference can be observed by comparing the results of the stereo setups with baselines of 0.416 m and 0.756 m. Since matching requires the cameras to be as parallel as possible, we can state that there is no need to increase the baseline of our stereo setup above 0.4 m, since it does not provide any improvement in reconstruction accuracy and it would make the matching process more difficult.

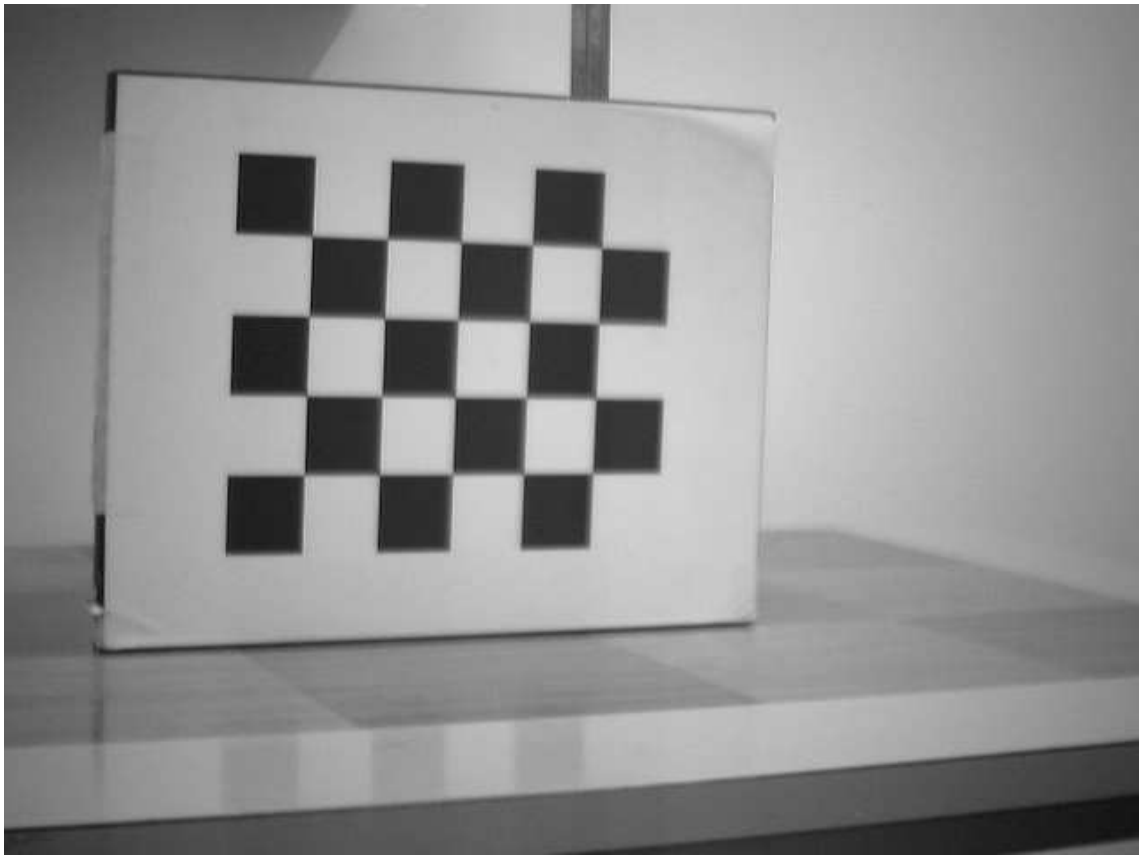


Figure 2.6: Calibration pattern

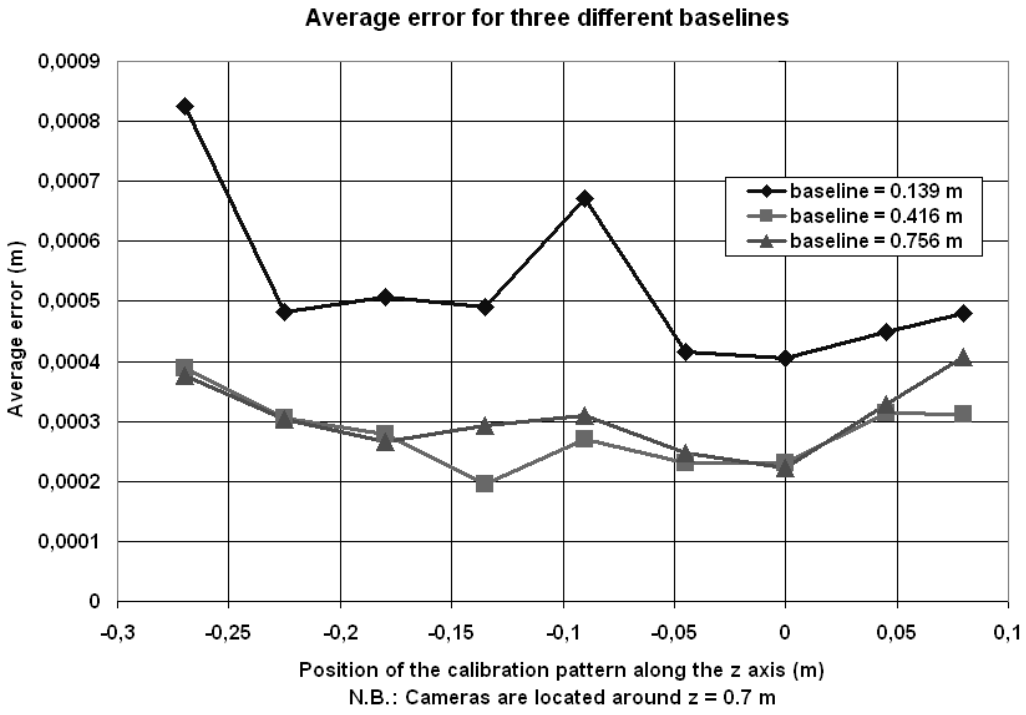


Figure 2.7: Average reconstruction error over a plane of 20 feature points, for three different baselines

Chapter 3

Stereo Setup Calibration

The calibration of a stereo setup is a critical step in any computer vision application where physical measurements have to be performed. The intrinsic and extrinsic calibration parameters of the cameras are the settings of the model that need to be estimated. If they are accurate, so will be the reconstruction data. If they are approximate, so will be the reconstruction data. Among the approaches existing in the literature, many involve direct measurements of the physical camera parameters. For example, one could measure the relative position between two cameras, and their relative orientation in order to figure out the homogeneous transformation that links them. In the scope of this thesis, we will not investigate these methods. The only physical measurements that will need to be performed will be on a *calibration pattern*, which is a pattern of regularly spaced and easy to detect feature points. Determining the position of the feature points on a calibration pattern is a straightforward procedure, and can be done with high precision. Furthermore, it will be assumed that the calibration pattern can be positioned in a global reference system with sufficient precision. No angular measurements will be performed. Starting with the basic information of the 3D location of feature points on a calibration pattern, along with the corresponding image points in pixels, we will be processing the data to compute the calibration parameters.

We will be using an intrinsic calibration matrix built with 4 parameters, as seen

in equation (2.3). Many authors include a 5^{th} intrinsic parameter, the *skew factor*, γ:

$$K(\frac{f}{l_x}, \frac{f}{l_y}, c_x, c_y, \gamma) = \begin{bmatrix} \frac{f}{l_x} & \gamma & c_x \\ 0 & \frac{f}{l_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The skew factor accounts for any non-orthogonality between the axes of the detector array. It is typically a small factor with respect to $\frac{f}{l_x}$ and $\frac{f}{l_y}$ (in other words, the array of detectors of most cameras is very close to being orthogonal). It has been observed that the introduction of this parameter does not improve significantly the calibration quality and, in some instances, may induce computation instability. For these reasons, we chose to neglect this effect by setting $\gamma = 0$.

The extrinsic calibration parameters are the rotation and the translation linking a camera's reference frame with a global reference frame. Combining the rotation and the translation in a single 4×4 matrix gives the extrinsic calibration matrix $Q_{cam/world}$ that obeys the convention of (A.46):

$$\vec{X}|_{world} = Q_{cam/world}\vec{X}|_{cam} \tag{3.2}$$

3.1 Single Camera Calibration

We decided to implement a calibration algorithm ourselves, after having tried software packages that did not yield repeatable results. Since intrinsic calibration parameters are independent of the camera pose, repetitions of the calibration procedure on different stereo setup must provide similar intrinsic parameters.

The calibration of a single camera has been the object of extensive research. In [4], Tsai argues that any stereo setup whose cameras are linearly modelled suffers important 3D reconstruction error. This postulate justifies the need to take radial distortion into account. A two-stage calibration technique is described that allows the computation of the extrinsic calibration parameters, the focal length of the camera

and the first order coefficient of the radial distortion phenomenon. The principal point and the pixel sizes are assumed to be known. The first stage exploits the *radial alignment constraint*, i.e. the fact that a distorted image point must lie on a line defined by the principal point and the undistorted image point. The radial alignment constraint is a set of equations allowing the computation of R, T_x and T_y. The second stage involves an iterative method to compute f, T_z and κ (the first order distortion parameter). The initial guess is computed linearly by neglecting lens distortion. This approach uses known 3D points and their image locations.

In [5], Zhang proposes a single camera calibration technique that exploits homographies between different views of a planar calibration pattern. This method takes radial distortion into account by computing the two first order coefficients. The homographies allow the computation of the intrinsic calibration matrix K (which includes the skewness parameter γ). The extrinsic calibration parameters can then be calculated, with respect to the locations of the calibration patterns. Finally, all the calibration parameters are refined through an iterative method. The advantage of this method is the fact that all the calibration patterns don't need to be precisely positioned: the 3D points that are needed are the relative position of the feature points on the calibration pattern. The extrinsic calibration parameters that are returned are relative to the locations of the calibration patterns, one of which may be taken as the world reference frame.

In [10], Seo and Hong propose a method to calibrate a linear camera that is allowed to zoom and rotate around its center of projection, but not to translate. This situation corresponds to a rotating and zooming camera mounted on a tripod, observing a scene that is far away. The calibration can then be used to build a mosaic of images. The method is based on the computation of the homography between images from matching points on a planar surface.

Artificial intelligence tools have also inspired many researchers working on camera calibration, especially genetic algorithms ([42], [43]).

The simplest method to achieve single camera calibration linearly is described in

[49]. It is composed of two steps:

1. Estimate the projection matrix
2. Decompose the projection matrix to retrieve K and $Q_{cam/world}$

3.1.1 Estimation of the Projection Matrix

Let us assume we have a set of n 3D points for which we know the global homogeneous coordinates $\vec{X}_i$. Each point, along with its corresponding image coordinates $\vec{u}_i$, allows one to write an equation of the form (2.10):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_i = \lambda_i \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_i \tag{3.3}$$

where p_{ij} represents entry (i, j) of the projection matrix P. Eliminating the λ_i from the expressions yields the following pair of linear equations:

$$p_{00}x_i + p_{01}y_i + p_{02}z_i + p_{03} - p_{20}u_ix_i - p_{21}u_iy_i - p_{22}u_iz_i - p_{23}u_i = 0 \tag{3.4}$$

$$p_{10}x_i + p_{11}y_i + p_{12}z_i + p_{13} - p_{20}v_ix_i - p_{21}v_iy_i - p_{22}v_iz_i - p_{23}v_i = 0 \tag{3.5}$$

Putting together the information of the n 3D points ($n \geq 6$) gives $2n$ linear equations in the 12 unknowns $p_{00}, p_{01}, ..., p_{23}$:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & -u_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & -v_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & -u_2 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2 & -v_2y_2 & -v_2z_2 & -v_2 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_nx_n & -u_ny_n & -u_nz_n & -u_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -v_nx_n & -v_ny_n & -v_nz_n & -v_n \end{bmatrix} \begin{bmatrix} p_{00} \\ p_{01} \\ p_{02} \\ p_{03} \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{20} \\ p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ ... \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

Since the projection matrix P is defined up to a scale factor, eleven of the twelve unknowns can be found as the non-trivial solution of this homogeneous equation.

3.1.2 Decomposition of the Projection Matrix

This method is described in [1], [50]. The explicit form of the projection matrix is, from (2.11):

$$
\begin{aligned}
P &= K[I|0]Q^{-1}_{cam/world} && (3.7)\\
p_{00} &= r_{00}\frac{f}{l_x} + r_{02}c_x && (3.8)\\
p_{01} &= r_{10}\frac{f}{l_x} + r_{12}c_x && (3.9)\\
p_{02} &= r_{20}\frac{f}{l_x} + r_{22}c_x && (3.10)\\
p_{03} &= -\frac{f}{l_x}(r_{00}T_x + r_{10}T_y + r_{20}T_z) - c_x(r_{02}T_x + r_{12}T_y + r_{22}T_z) && (3.11)\\
p_{10} &= r_{01}\frac{f}{l_y} + r_{02}c_y && (3.12)\\
p_{11} &= r_{11}\frac{f}{l_y} + r_{12}c_y && (3.13)\\
p_{12} &= r_{21}\frac{f}{l_y} + r_{22}c_y && (3.14)\\
p_{13} &= -\frac{f}{l_y}(r_{01}T_x + r_{11}T_y + r_{21}T_z) - c_y(r_{02}T_x + r_{12}T_y + r_{22}T_z) && (3.15)\\
p_{20} &= r_{02} && (3.16)\\
p_{21} &= r_{12} && (3.17)\\
p_{22} &= r_{22} && (3.18)\\
p_{23} &= -r_{02}T_x - r_{12}T_y - r_{22}T_z && (3.19)
\end{aligned}
$$

From equations (3.16) to (3.18), it can be seen that the third column of the rotation matrix is directly available from inspection of the first three entries of the third row of P. At this point, a scale factor must be calculated such that the sum $r_{02}^2+r_{12}^2+r_{22}^2 = 1$, from the orthogonality property (A.16). The scale factor $\frac{1}{\sqrt{p_{20}^2+p_{21}^2+p_{22}^2}}$ is applied to the projection matrix before any further computation.

Let us define the three vectors [1]:

$$\vec{q_1} \equiv \begin{bmatrix} p_{00} \\ p_{01} \\ p_{02} \end{bmatrix} = \begin{bmatrix} r_{00}\frac{f}{l_x} + r_{02}c_x \\ r_{10}\frac{f}{l_x} + r_{12}c_x \\ r_{20}\frac{f}{l_x} + r_{22}c_x \end{bmatrix} \tag{3.20}$$

$$\vec{q_2} \equiv \begin{bmatrix} p_{10} \\ p_{11} \\ p_{12} \end{bmatrix} = \begin{bmatrix} r_{01}\frac{f}{l_y} + r_{02}c_y \\ r_{11}\frac{f}{l_y} + r_{12}c_y \\ r_{21}\frac{f}{l_y} + r_{22}c_y \end{bmatrix} \tag{3.21}$$

$$\vec{q_3} \equiv \begin{bmatrix} p_{20} \\ p_{21} \\ p_{22} \end{bmatrix} = \begin{bmatrix} r_{02} \\ r_{12} \\ r_{22} \end{bmatrix} \tag{3.22}$$

By making use of the orthogonality properties of the rotation matrix (A.15) and (A.16), the following equations can be derived, expressing most of the remaining calibration parameters:

$$c_x = \vec{q_1} \cdot \vec{q_3} \tag{3.23}$$

$$c_y = \vec{q_2} \cdot \vec{q_3} \tag{3.24}$$

$$\frac{f}{l_x} = \sqrt{\vec{q_1} \cdot \vec{q_1} - c_x^2} \tag{3.25}$$

$$\frac{f}{l_y} = \sqrt{\vec{q_2} \cdot \vec{q_2} - c_y^2} \tag{3.26}$$

$$r_{00} = \frac{p_{00} - r_{02}c_x}{\frac{f}{l_x}} \tag{3.27}$$

$$r_{01} = \frac{p_{10} - r_{02}c_y}{\frac{f}{l_y}} \tag{3.28}$$

$$r_{10} = \frac{p_{01} - r_{12}c_x}{\frac{f}{l_x}} \tag{3.29}$$

$$r_{11} = \frac{p_{11} - r_{12}c_y}{\frac{f}{l_y}} \tag{3.30}$$

$$r_{20} = \frac{p_{02} - r_{22}c_x}{\frac{f}{l_x}} \tag{3.31}$$

$$r_{21} = \frac{p_{12} - r_{22}c_y}{\frac{f}{l_y}} \tag{3.32}$$

In a typical situation, the obtained rotation matrix will not be perfectly orthogonal. Orthogonality must be enforced, e.g. through singular value decomposition. If the obtained first estimate of the rotation matrix is $R' = UD'V^T$, then the orthogonal matrix R that is the closest to R' in the sense of the Frobenius norm will be computed by replacing D' by an identity matrix, i.e. $R = UV^T$.

The three last parameters T_x, T_y and T_z can be retrieved by solving a system of three linear equations in three unknowns, given by the expressions of p_{03}, p_{13} and p_{23}.

3.2 Calibration of a Pair of Cameras

In [9], Ramanathan et al. propose a method to refine the knowledge of the extrinsic calibration parameters, from an estimate of the location of the views. The intrinsic calibration parameters are assumed to be known. The main idea is based on the projection of the silhouette cone of an object, as defined by another view. In case of perfect knowledge of the cameras locations, the projected cone would be a triangle that is tangent to the object surface as seen in the image of interest. In case of imperfect knowledge of the cameras poses, the triangle will not be perfectly tangent to the object silhouette. A measure of mismatch is defined, and an error gradient is monitored as the extrinsic calibration parameters of both cameras are perturbed. The procedure is repeated iteratively on all the possible pairs of views.

In [6], Malm and Heyden use the single camera procedure described by Zhang in [5] and extend it to a stereo setup, with the exception that radial distortion is not considered. The availability of two cameras provides additional constraints, obtained by arbitrarily setting the position of the planar calibration pattern in the plane $Z = 0$. This special configuration leads to a homography between the pixel coordinates of the image points and the (X, Y) coordinates of the calibration pattern feature points. Using images from a minimum of two positions of the stereoscopic setup, the homographies are built and their relationship is used to compute the intrinsic calibration

matrices. In a second step, the homogeneous transformation matrix linking the two cameras is obtained. This method is claimed to be robust to noise and especially efficient when one of the cameras is better than the other.

Memon and Khan [16] approached the stereo calibration problem in an original way. Instead of trying to compute calibration parameters, they trained a neural network to perform 3D reconstruction from a match of image points. They precisely positioned a calibration pattern with respect to a global reference system, and recorded the image points corresponding to the feature points. They supplied the data to a neural network for training. Each training data had, as input, the matched image coordinates of a feature point and, as output, its 3D coordinates. The neural network could learn to perform 3D reconstruction, but nothing was known about the actual values of the calibration parameters.

Do [17] exploits basically the same idea, with the difference that his neural network has the task of computing the difference between the image coordinates predicted by a linear model and the actual observed image coordinates.

Additional research topics on stereo calibration can be found in the literature ([45], [46], [47], [48]).

One possible new approach is to take advantage of the fundamental matrix, which might provide additional information about intrinsic and extrinsic calibration parameters. It is computable from matches, as described in Section 2.4.2. Those matches can be obtained by making use of a calibration pattern exhibiting features that are easily identified.

3.2.1 Two Ways to Compute the Fundamental Matrix

Let us assume the two cameras have been calibrated independently through the procedure described in Sections 3.1.1 and 3.1.2, providing K_1, K_2, $Q_{cam1/world}$ and

$Q_{cam2/world}$. The homogeneous transformation linking the two cameras can be computed:

$$Q_{cam2/cam1} = Q^{-1}_{cam1/world} Q_{cam2/world} \tag{3.33}$$

Then, the fundamental matrix can be built, from (2.15):

$$F = K_2^{-T} R^T_{cam2/cam1} S_{cam2/cam1} K_1^{-1} \tag{3.34}$$

As seen in Section 2.4.2, the fundamental matrix can also be estimated from matches in two views (eight-point algorithm). Let us suppose the matrix obtained by the eight-point algorithm differs substantially from the one obtained by independent calibration of the two cameras (3.34). Which one should we trust? The remainder of this chapter will be used to determine which is the best approach.

The main problem associated with the fundamental matrix obtained from the eight-point algorithm is the potential inconsistency with the intrinsic calibration parameters. Keeping only F and the intrinsic calibration matrices K_1 and K_2, from (2.14) and (2.15), the essential matrix can be computed:

$$E = R^T_{cam2/cam1} S_{cam2/cam1} = K_2^T F K_1 \tag{3.35}$$

The rotation $R_{cam2/cam1}$ and the translation $\vec{T}_{cam2/cam1}$ can be retrieved through the decomposition of the essential matrix (cf. Section 3.2.2). It will be shown that the expression of E obtained from (3.35), using F from the eight-point algorithm may lead to a matrix that cannot be a valid essential matrix.

3.2.2 Decomposition of the Essential Matrix

This technique is described by Longuet-Higgins in [2]. The first observation that can be done about the essential matrix is the fact that the product of its transpose by

itself is solely dependent on the translation vector entries:

$$\begin{aligned} E^TE &= (R^TS)^T \cdot R^TS = S^TRR^TS \overset{(A.12)}{=} S^TRR^{-1}S = S^TS \\ &\overset{(2.16)}{=} \begin{bmatrix} 0 & T_z & -T_y \\ -T_z & 0 & T_x \\ T_y & -T_x & 0 \end{bmatrix} \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \\ &= \begin{bmatrix} T_y^2+T_z^2 & -T_xT_y & -T_xT_z \\ -T_xT_y & T_x^2+T_z^2 & -T_yT_z \\ -T_xT_z & -T_yT_z & T_x^2+T_y^2 \end{bmatrix} \end{aligned} \tag{3.36}$$

The trace of E^TE will provide us with the scale factor such that the length of the translation vector is normalized to 1. This is allowed since the matrix S has rank 2 (its eigenvalues are $0, \pm i\sqrt{T_x^2+T_y^2+T_z^2}$) and therefore is defined up to a scale factor.

$$trace(E^TE) = 2T_x^2 + 2T_y^2 + 2T_z^2 = 2|\vec{T}|^2 \tag{3.37}$$

$$N \equiv |\vec{T}| = \sqrt{\frac{trace(E^TE)}{2}} \tag{3.38}$$

The $\hat{S}$ (normalized S) and $\hat{E}$ (normalized E) matrices can be defined:

$$\hat{S} \equiv \frac{1}{N}S \tag{3.39}$$

$$\hat{E} \equiv \frac{1}{N}E = R^T\hat{S} \tag{3.40}$$

$$\begin{aligned} \hat{E}^T\hat{E} &= \hat{S}^T\hat{S} \\ &= \begin{bmatrix} 1-\hat{T_x}^2 & -\hat{T_x}\hat{T_y} & -\hat{T_x}\hat{T_z} \\ -\hat{T_x}\hat{T_y} & 1-\hat{T_y}^2 & -\hat{T_y}\hat{T_z} \\ -\hat{T_x}\hat{T_z} & -\hat{T_y}\hat{T_z} & 1-\hat{T_z}^2 \end{bmatrix} \end{aligned} \tag{3.41}$$

From inspection of (3.41), the normalized components of the translation vector, $\hat{T_x}$, $\hat{T_y}$ and $\hat{T_z}$ can be evaluated. Notice that some unrealistic situations can arise:

- entries on the main diagonal of $\hat{E}^T\hat{E}$ can be outside the range $[0, 1]$
- the redundancy of information can be contradictory

This is where the discrepancy between the fundamental matrix and the intrinsic calibration matrices obtained independently becomes apparent.

The matrix $\hat{E}^T\hat{E}$ has rank 2 (its eigenvalues are 0, 1, 1). Its singular value decomposition should return a diagonal matrix with singular values 1, 1 and 0. In practice, it might not be the case, as just pointed out. If SVD returns $\hat{E}^T\hat{E}_{initial} = UD'V^T$ such that the diagonal matrix D' has two entries slightly off unity and one entry slightly off 0, it should be replaced by another diagonal matrix D with two entries which are exactly one and one null entry. Then, $\hat{E}^T\hat{E}_{corrected} = UDV^T$ is the valid matrix closest to $\hat{E}^T\hat{E}_{initial}$ in the sense of the Frobenius norm.

For now, it is important to recognize the ambiguity on the signs of the components, that comes from the quadratic terms on the main diagonal of $\hat{E}^T\hat{E}$. It will be resolved by first assuming a given sign combination, and confirm or prove wrong the assumption later on:

- choose one of the non-zero components ($\hat{T}_x$ for instance) and assume its sign is positive
- compute the remaining components in accordance with the sign assumption

We will define $\vec{c_i}$ as the vector whose entries are the i^{th} column of the rotation matrix $R_{cam2/cam1}$. From (A.26):

$$\vec{c_\alpha} = \vec{c_\beta} \times \vec{c_\gamma} \tag{3.42}$$

where (α, β, γ) is an even permutation of $(1, 2, 3)$.

By direct computation of the entries of $\hat{E} = R^T\hat{S}$, it can be shown that the vector $\vec{\hat{e}_i}$ built with the entries of the i^{th} row of $\hat{E}$ obeys to the following identity:

$$\vec{\hat{e}_i} = \vec{c_i} \times \vec{\hat{T}} \tag{3.43}$$

Let us define the vector $\vec{w_i}$ as the cross product of $\vec{\hat{T}}$ and $\vec{\hat{e}_i}$:

$$\vec{w_i} \equiv \vec{\hat{T}} \times \vec{\hat{e}_i} \tag{3.44}$$

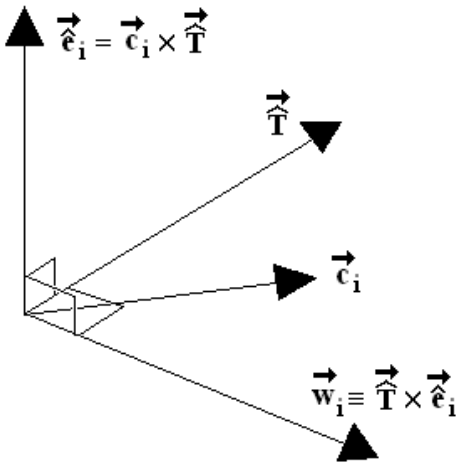


Figure 3.1: Vectors defined for the decomposition of the essential matrix

Since $\vec{c_i}$ lies in the plane defined by the vectors $\vec{\hat{T}}$ and $\vec{w_i}$ (see Figure 3.1), it can be expressed by a linear combination of these two orthogonal vectors:

$$\vec{c_i} = a_i \vec{\hat{T}} + b_i \vec{w_i} \tag{3.45}$$

Inserting (3.45) into (3.43):

$$\begin{aligned} \vec{\hat{e}_i} &= (a_i \vec{\hat{T}} + b_i \vec{w_i}) \times \vec{\hat{T}} && (3.46) \\ &= b_i \vec{w_i} \times \vec{\hat{T}} && (3.47) \\ &\overset{(3.44)}{=} b_i \vec{\hat{T}} \times \vec{\hat{e}_i} \times \vec{\hat{T}} && (3.48) \end{aligned}$$

Since the modulus of $\vec{\hat{T}}$ is 1 and $\vec{\hat{T}}$ is perpendicular to $\vec{\hat{e}_i}$ (3.43), $\vec{\hat{T}} \times \vec{\hat{e}_i} \times \vec{\hat{T}} = \vec{\hat{e}_i}$ (see Figure 3.2) and

$$b_i = 1 \tag{3.49}$$

Inserting (3.49) and (3.45) into (3.42):

$$\begin{aligned} a_\alpha \vec{\hat{T}} + \vec{w_\alpha} &= (a_\beta \vec{\hat{T}} + \vec{w_\beta}) \times (a_\gamma \vec{\hat{T}} + \vec{w_\gamma}) && (3.50) \\ a_\alpha \vec{\hat{T}} + \vec{w_\alpha} &= a_\beta \vec{\hat{T}} \times \vec{w_\gamma} + a_\gamma \vec{w_\beta} \times \vec{\hat{T}} + \vec{w_\beta} \times \vec{w_\gamma} && (3.51) \end{aligned}$$

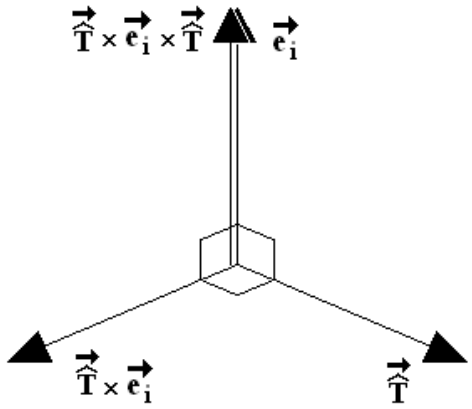


Figure 3.2: $\vec{\hat{T}} \times \vec{\hat{e}_i} \times \vec{\hat{T}}$

From (3.44), $\vec{w_i}$ is orthogonal to $\vec{\hat{T}}$. Hence, we can equalize the first term of the left-hand side with the third term on the right-hand side of (3.51):

$$a_\alpha \vec{\hat{T}} = \vec{w_\beta} \times \vec{w_\gamma} \tag{3.52}$$

Inserting (3.49) and (3.52) into (3.42) gives the central result of the essential matrix decomposition:

$$\vec{c_\alpha} = \vec{w_\alpha} + \vec{w_\beta} \times \vec{w_\gamma} \tag{3.53}$$

where (α, β, γ) is an even permutation of $(1, 2, 3)$. Equation (3.53) allows us to compute the rotation matrix columns from the normalized translation vector $\vec{\hat{T}}$ and the normalized essential matrix $\hat{E}$.

At this point, since K_1, K_2, $R_{cam2/cam1}$ and $\vec{\hat{T}}_{cam2/cam1}$ are known, 3D reconstruction up to a scale factor can be performed using the techniques of Section 2.5. In order to resolve the ambiguity regarding the signs combination of the translation components, it is necessary to perform 3D reconstruction on a pair of matching points. The 3D location of any point seen by the cameras, with respect to the reference system

of camera 1, must have positive Z-coordinate. If the computed Z-coordinate is negative, this is an indication that the assumed sign for one of the non-zero components was wrong. Therefore, it must be negated and the rest of the computations must be redone from this point.

The decomposition of the essential matrix supplies the translation vector normalized to a unity norm. In order to get a translation vector consistent with the distance units of the coordinates systems (in meters, for example), it is necessary to *denormalize* the translation vector. The scale factor can be obtained by computing the ratio of the absolute distance between two points and their normalized distance, as computed with the normalized translation vector. Of course, it is better to use a large number of measurements, and average out. The obtained scale factor is multiplied with the normalized translation vector, and the homogeneous transformation matrix between camera 1 and camera 2 can be built.

3.2.3 Least-Square Refinement of the Intrinsic Calibration Parameters

The observation of a case where the SVD of $\hat{E}^T\hat{E}$ doesn't lead to singular values of (1, 1, 0) means that something is wrong in the initial estimates. In practice, it is frequently the case when we combine intrinsic calibration matrices obtained independently with a fundamental matrix obtained through the eight-point algorithm. The observation of an inconsistency in $\hat{E}^T\hat{E}$ means that either the intrinsic calibration matrices or the fundamental matrix are inaccurate. We aim at comparing the validity of the fundamental matrix as obtained through the eight-point algorithm with the one computed after independent calibration of the cameras. As a consequence, we will interpret this inconsistency as a result of inaccuracies in the intrinsic calibration matrices.

We will reconciliate our intrinsic matrices with the fundamental matrix through an iterative refinement technique. We will aim at minimizing the reprojection error.

Starting from known 3D points, we will be searching for the intrinsic parameters of both cameras such that the 3D points are projected as close as possible to their observed respective image points.

We will assume the homogeneous transformation matrices between the global reference frame and the cameras are known. This is where the iterative procedure comes into play: we start with an estimate of the intrinsic parameters to compute an estimate of the extrinsic parameters, that in turn are used to compute a better estimate of the intrinsic parameters, and so on. Once again, the mathematical treatment will start with (2.10):

$$u_m = \lambda_m P \vec{X}_m \tag{3.54}$$

$$\begin{bmatrix} u_m \\ v_m \\ 1 \end{bmatrix} = \lambda_m \begin{bmatrix} \frac{f}{l_x} & 0 & c_x \\ 0 & \frac{f}{l_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} Q^{-1} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}_{measured} \tag{3.55}$$

Let us define the known constants $a, b, c, ..., l$ that are computed from:

$$[I_3|0]Q^{-1} \equiv \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \tag{3.56}$$

$$\begin{bmatrix} u_m \\ v_m \\ 1 \end{bmatrix} = \lambda_m \begin{bmatrix} \frac{f}{l_x} & 0 & c_x \\ 0 & \frac{f}{l_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}_{measured} \tag{3.57}$$

$$\begin{bmatrix} u_m \\ v_m \\ 1 \end{bmatrix} = \lambda_m \begin{bmatrix} a\frac{f}{l_x} + ic_x & b\frac{f}{l_x} + jc_x & c\frac{f}{l_x} + kc_x & d\frac{f}{l_x} + lc_x \\ e\frac{f}{l_y} + ic_y & f\frac{f}{l_y} + jc_y & g\frac{f}{l_y} + kc_y & h\frac{f}{l_y} + lc_y \\ i & j & k & l \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{3.58}$$

We can get rid of the non-linearity by evaluating λ_m, for each 3D point:

$$\lambda_m = \frac{1}{iX_m + jY_m + kZ_m + l} \tag{3.59}$$

Let us assume we have two cameras. In this case, we will denote $\lambda_{p,m}$ the λ-factor obtained with camera p ($p = \{1, 2\}$) and feature point m. Inserting (3.59) into (3.58) gives an over-determined linear system with $4n$ equations in 8 unknowns, the intrinsic calibration parameters of the two cameras.

$$\kappa \begin{bmatrix} \frac{f}{l_x}1 \\ \frac{f}{l_y}1 \\ c_{x1} \\ c_{y1} \\ \frac{f}{l_x}2 \\ \frac{f}{l_y}2 \\ c_{x2} \\ c_{y2} \end{bmatrix} = \begin{bmatrix} u_{1,1} \\ v_{1,1} \\ u_{2,1} \\ v_{2,1} \\ u_{1,2} \\ v_{1,2} \\ u_{2,2} \\ v_{2,2} \\ ... \\ u_{1,n} \\ v_{1,n} \\ u_{2,n} \\ v_{2,n} \end{bmatrix} \tag{3.60}$$

where, for $m = \{0, 1, 2, ..., n-1\}$, $\vec{\kappa}_m$ being the m^{th} row of κ:

$$\begin{aligned}
\vec{\kappa}_{4m} &= \begin{bmatrix} \lambda_{1,m}(a_1X_m + b_1Y_m + c_1Z_m + d_1) & 0 & \lambda_{1,m}(i_1X_m + j_1Y_m + k_1Z_m + l_1) & \mathbf{0}_{1\times5} \end{bmatrix} \\
\vec{\kappa}_{4m+1} &= \begin{bmatrix} 0 & \lambda_{1,m}(e_1X_m + f_1Y_m + g_1Z_m + h_1) & 0 & \lambda_{1,m}(i_1X_m + j_1Y_m + k_1Z_m + l_1) & \mathbf{0}_{1\times4} \end{bmatrix} \\
\vec{\kappa}_{4m+2} &= \begin{bmatrix} \mathbf{0}_{1\times4} & \lambda_{2,m}(a_2X_m + b_2Y_m + c_2Z_m + d_2) & 0 & \lambda_{2,m}(i_2X_m + j_2Y_m + k_2Z_m + l_2) & 0 \end{bmatrix} \\
\vec{\kappa}_{4m+3} &= \begin{bmatrix} \mathbf{0}_{1\times5} & \lambda_{2,m}(e_2X_m + f_2Y_m + g_2Z_m + h_2) & 0 & \lambda_{2,m}(i_2X_m + j_2Y_m + k_2Z_m + l_2) \end{bmatrix}
\end{aligned}$$

Solving (3.60) through a least-square error technique will minimize $(u_{1,1_{measured}} - u_{1,1_{reprojected}})^2 + (v_{1,1_{measured}} - v_{1,1_{reprojected}})^2 + ... + (v_{2,n_{measured}} - v_{2,n_{reprojected}})^2$. In other words, the reprojection error will be minimized.

3.2.4 Iterative Algorithm for Stereo Setup Calibration

We now have all the mathematical tools to describe the iterative algorithm that modifies the intrinsic calibration parameters of a two-camera stereo setup, to make them consistent with the fundamental matrix obtained through the eight-point algorithm.

- Input data:
  - An estimate of the intrinsic calibration matrices K_1 and K_2;
  - A fundamental matrix F obtained from a large number of good matches($\sim$ 300);
  - A large number of 3D points ($\sim$ 200) with their corresponding image points in both images.
- Output data: Projection matrices P_1 and P_2 that are in agreement with F.

1. From F and the actual estimates of K_1 and K_2, compute the essential matrix: $E = K_2^T F K_1$.
2. Decompose E, through the technique described in section 3.2.2 to get $R_{cam2/cam1}$ and $\hat{\vec{T}}_{cam2/cam1}$.
3. Perform 3D reconstruction of the 3D points using their pairs of image points through the technique described in section 2.5 to get the normalized calculated 3D locations, with respect to camera 1.
4. Denormalize the 3D locations to get the modulus of the translation vector.

5. Register the cloud of denormalized 3D points with the cloud of measured 3D points through the technique of section A.4 to get the homogeneous transformation linking the first camera with the global reference frame, $Q_{cam1/world}$.

6. From $Q_{cam1/world}$, $R_{cam2/cam1}$ and $\vec{T}_{cam2/cam1}$, compute the homogeneous transformation linking camera 2 with the global reference frame, $Q_{cam2/world}$.

7. Compute the new estimates of P_1 and P_2 through equation (2.11).

8. From $Q_{cam1/world}$, $Q_{cam2/world}$, the measured 3D points and their observed image points, refine the intrinsic calibration parameters through least-square solving of (3.60). Better estimates of K_1 and K_2 are obtained.

9. If the reprojection error is sufficiently small, exit. Otherwise, go back to step 1.

3.3 Experimental Comparison Between Calibration of a Pair of Cameras Independently and the Iterative Algorithm

3.3.1 Repeatability of the Fundamental Matrix and the Projection Matrices

A stereo setup has been calibrated twice to observe the repeatability of both methods. The subscript a refers to the first set of data while the subscript b refers to the second set of data. The numerical subscripts 1 and 2 will refer to the first and the second camera, respectively. The fundamental matrices $F(a)$ and $F(b)$ are given in Table 3.1.

These two fundamental matrices were computed with the eight-point algorithm, using 600 matches. Despite the large number of matches, significant variation can be observed between the corresponding entries of the two matrices.

Table 3.1: Comparison of two fundamental matrices obtained through a repetition of the eight-point algorithm on the same stereo setup

Data set	F
$F(a)$	$\begin{bmatrix} 1 & -5.35854 & -2953.42 \\ -9.62767 & -1.27071 & 61249.6 \\ 6184.38 & -57162 & -3.35226e+006 \end{bmatrix}$
$F(b)$	$\begin{bmatrix} 1 & -4.72328 & -2940.52 \\ -10.4527 & -0.41363 & 59487.8 \\ 6244.39 & -55604.9 & -3.2822e+006 \end{bmatrix}$

Table 3.2: Comparison of the projection matrices of the two cameras, obtained through a repetition of the manipulation on the same stereo setup (method of Section 3.1.1)

Data set	P
$P_1(a)$	$\begin{bmatrix} 1280.18 & 19.4154 & -174.738 & 375.071 \\ 81.8363 & -1240.17 & -12.5186 & 318.709 \\ 0.168546 & -0.152118 & -0.973885 & 0.772105 \end{bmatrix}$
$P_1(b)$	$\begin{bmatrix} 1280.6 & 19.8226 & -174.772 & 375.092 \\ 81.7831 & -1240.42 & -12.476 & 318.77 \\ 0.16843 & -0.151046 & -0.974072 & 0.772125 \end{bmatrix}$
$P_2(a)$	$\begin{bmatrix} 1163.23 & -40.0977 & -566.029 & 244.562 \\ -12.3381 & -1269.95 & -74.4566 & 337.468 \\ -0.212123 & -0.139376 & -0.967253 & 0.740569 \end{bmatrix}$
$P_2(b)$	$\begin{bmatrix} 1165.58 & -39.8649 & -566.936 & 244.986 \\ -12.1115 & -1272.12 & -74.0658 & 338.13 \\ -0.211713 & -0.138431 & -0.967479 & 0.741964 \end{bmatrix}$

Table 3.3: Comparison of the intrinsic calibration matrices of the cameras, obtained from decomposition of the initial projection matrices of Table 3.2.

Data set	K
$K_1(a)$	$\begin{bmatrix} 1234.14 & 0 & 382.991 \\ 0 & 1224.26 & 214.637 \\ 0 & 0 & 1 \end{bmatrix}$
$K_1(b)$	$\begin{bmatrix} 1234.6 & 0 & 382.938 \\ 0 & 1224.74 & 213.288 \\ 0 & 0 & 1 \end{bmatrix}$
$K_2(a)$	$\begin{bmatrix} 1257.48 & 0 & 306.334 \\ 0 & 1247.06 & 251.636 \\ 0 & 0 & 1 \end{bmatrix}$
$K_2(b)$	$\begin{bmatrix} 1259.83 & 0 & 307.248 \\ 0 & 1249.5 & 250.322 \\ 0 & 0 & 1 \end{bmatrix}$

As can be seen from Table 3.2, the projection matrices obtained through a repetition of the manipulations of Section 3.1.1 are very similar to each other. As a consequence, the intrinsic calibration matrices that can be extracted from these projection matrices are similar (cf. Table 3.3).

3.3.2 Repeatability of the Intrinsic Calibration Matrices Obtained through Iterative Refinement Versus Decomposition of the Initial Projection Matrices

The fundamental matrices $F(a)$ and $F(b)$ as shown in Table 3.1 were used to refine the initial intrinsic calibration matrices, in order to make them consistent with each other. The obtained intrinsic calibration matrices are shown in Table 3.4.

Since the initial intrinsic calibration matrices used to seed the iterative refinement algorithm were very similar (cf. Table 3.3), the slight discrepancies observed in the final intrinsic calibration matrices shown in Table 3.4 can only be explained in terms of the differences in the computed fundamental matrices. In other words, the relatively poor repeatability of the fundamental matrix obtained from the eight-point algorithm

Table 3.4: Comparison of the intrinsic calibration matrices of the two cameras, after iterative refinement to make them consistent with the fundamental matrices of Table 3.1.

Data set	K
$K_1(a)$	$\begin{bmatrix} 1296.77 & 0 & 352.238 \\ 0 & 1285.61 & 190.565 \\ 0 & 0 & 1 \end{bmatrix}$
$K_1(b)$	$\begin{bmatrix} 1284.91 & 0 & 358.397 \\ 0 & 1273.93 & 184.015 \\ 0 & 0 & 1 \end{bmatrix}$
$K_2(a)$	$\begin{bmatrix} 1313.03 & 0 & 340.128 \\ 0 & 1302.6 & 224.831 \\ 0 & 0 & 1 \end{bmatrix}$
$K_2(b)$	$\begin{bmatrix} 1291.92 & 0 & 332.384 \\ 0 & 1282 & 242.497 \\ 0 & 0 & 1 \end{bmatrix}$

has a negative impact on the repeatability of the intrinsic calibration matrices, in the case of the iterative refinement method of Section 3.2.4. On the other hand, the repeatability of the projection matrices obtained with known 3D locations of feature points, as described in Section 3.1.1, is excellent.

3.3.3 Reprojection Error and Reconstruction Error

The parameters of interest in the context of evaluating the performance of a pair of projection matrices are the reprojection error and the reconstruction error. The reprojection error is the distance, in pixels, between the location of the image of a known 3D point and the location predicted by the projection matrix. The reconstruction error, for a stereo setup, is the distance, in meters, between the calculated location of a 3D point from the match coordinates and its actual location as physically measured.

We will compare the performance of the pairs of projection matrices of the two approaches. We will use the two sets of 3D points that were used to compute the initial projection matrices, for each comparison, in an attempt to reveal any overfitting that

Table 3.5: Comparison of the reprojection and reconstruction standard error obtained with the initial projection matrices versus the projection matrices obtained after iterative refinement of the intrinsic calibration matrices, for the first manipulation

	Set of points a	Set of points b
Initial $P_1(a)$ and $P_2(a)$	$\sigma_{reprojection} = 0.279 pixels$ $\sigma_{reconstruction} = 0.388 mm$	$\sigma_{reprojection} = 0.295 pixels$ $\sigma_{reconstruction} = 0.401 mm$
$P_1(a)$ and $P_2(a)$ after 500 refinement iterations	$\sigma_{reprojection} = 0.538 pixels$ $\sigma_{reconstruction} = 0.775 mm$	$\sigma_{reprojection} = 0.539 pixels$ $\sigma_{reconstruction} = 0.754 mm$

Table 3.6: Comparison of the reprojection and reconstruction standard error obtained with the initial projection matrices versus the projection matrices obtained after iterative refinement of the intrinsic calibration matrices, for the second manipulation

	Set of points a	Set of points b
Initial $P_1(b)$ and $P_2(b)$	$\sigma_{reprojection} = 0.285 pixels$ $\sigma_{reconstruction} = 0.398 mm$	$\sigma_{reprojection} = 0.295 pixels$ $\sigma_{reconstruction} = 0.393 mm$
$P_1(b)$ and $P_2(b)$ after 500 refinement iterations	$\sigma_{reprojection} = 0.433 pixels$ $\sigma_{reconstruction} = 0.652 mm$	$\sigma_{reprojection} = 0.435 pixels$ $\sigma_{reconstruction} = 0.627 mm$

could be present.

Comparing between the left and the right column of Tables 3.5 and 3.6 shows that the reconstruction results are not affected by the set of 3D points used, showing that there is no overfitting phenomenon that would strongly link a pair of projection matrices to the set of 3D points that was used to compute it. Comparing between the top and the bottom row of Tables 3.5 and 3.6 shows that the initial projection matrices perform better than the projection matrices resulting from iterative refinement of the intrinsic calibration matrices, both in terms of reprojection error and reconstruction error. This comes from the fact that, at each iteration, we perform 3D reconstruction using intrinsic calibration matrices that are corrupted by the error in the fundamental matrix. Although the error does decrease with the number of iteration, it never gets to the level obtained using only the initial projection matrices. Therefore, the process of reconciliating the intrinsic calibration matrices with the computed fundamental matrices leads to a significant loss in performance of the projection matrices. Now, in light of these experimental results, the question arises: Wouldn't it be better if,

Table 3.7: Comparison of the standard error of the distance between a feature point match and its associated epipolar line, using fundamental matrices computed through the eight-point algorithm versus computed by decomposition of the initial projection matrices, for the first manipulation

	$F(a)_{eight-point}$	$F(a)_{initialP's}$
Freely moving pattern a	0.306	0.634
Precisely positioned pattern a	0.237	0.146
Freely moving pattern b	0.292	0.425
Precisely positioned pattern b	0.241	0.152

instead, we would discard the fundamental matrix obtained through the eight-point algorithm (which seems to be somehow unrepeatable, as seen in section 3.3.1) and replace it by the one extracted from the initial projection matrices, as explained in Section 3.2? In terms of reconstruction, there would be a clear advantage, as just shown. The next question is: how would such a fundamental matrix compare with a fundamental matrix computed through the eight-point algorithm?

3.3.4 Comparison Between the Fundamental Matrix Obtained by Eight-Point Algorithm and the Fundamental Matrix Obtained by Decomposition of the Initial Projection Matrices

The task of the fundamental matrix is, given some feature point in one of the images, to provide an epipolar line along which the corresponding feature point must lie. Therefore, an appropriate performance measurement is the standard error in pixels between the actual location of a feature point match and its associated epipolar line. As opposed to the measurements displayed in Section 3.3.3 where 3D locations are needed, this comparison only requires 2D matches. As a consequence, we will be able to use sets of matches obtained with the calibration pattern moving freely, in addition to the sets of matches whose 3D locations are known.

Table 3.8: Comparison of the standard error (in pixels) of the distance between a feature point match and its associated epipolar line, using fundamental matrices computed through the eight-point algorithm versus computed by decomposition of the initial projection matrices, for the second manipulation

	$F(b)_{eight-point}$	$F(b)_{initialP's}$
Freely moving pattern a	0.346	0.622
Precisely positioned pattern a	0.214	0.152
Freely moving pattern b	0.262	0.418
Precisely positioned pattern b	0.218	0.156

Tables 3.7 and 3.8 show that the fundamental matrix computed through the eight-point algorithm performs better with the matches from a pattern moving freely in the volume of interest, while the fundamental matrix obtained through decomposition of the initial projection matrices performs better with the matches from the calibration pattern precisely located at known positions (the sets of points used to compute the initial projection matrices). This observation can be explained by recalling the linearity of the model. As we know, radial distortions become more and more important the further we get from the center of the image. The set of matches from a freely moving calibration pattern contains more matches in the periphery of the images, while the set of matches from the precisely positioned calibration pattern are concentrated in the center of the images. As a consequence, the fundamental matrix obtained through the eight-point algorithm does a better job at finding a compromise fundamental matrix that will allow the eccentric matches to be fairly well taken into account. On the other hand, the fundamental matrix computed through decomposition of the projection matrices started with matches in the linear region of the images, and performs better in this domain, but worse in the borders where non-linear effects get more important. In all cases, the measured standard errors are below 1 pixel. We cannot realistically expect improvement on this figure without introducing radial distortion in the model, which would increase significantly the complexity of calculation. To summarize, the comparison between the fundamental matrix obtained through the eight-point algorithm and the fundamental matrix obtained through decomposition

of the projection matrices is a tie match.

Typically, the work area of a stereo setup will be concentrated in the center of the image of the cameras. The justification is straightforward: points that lie far away from the center of an image have a high probability to be invisible by the other camera. This simple fact makes us confident that we don't need to worry about strongly distorted eccentric image points.

3.4 Discussion

Experimental results showed the superiority of the projection matrices obtained independently over the projection matrices resulting from the iterative refinement of the intrinsic calibration matrices, both in terms of reprojection error and reconstruction error. This is assumed to be due to the inaccuracy in the fundamental matrix computed through the eight-point algorithm, which corrupts the intrinsic calibration matrices when they are forced to be consistent with F through iterative refinement. It has been demonstrated that the fundamental matrix obtained through eight-point algorithm and the fundamental matrix obtained through decomposition of the initial projection matrices perform equally well, in terms of capacity to produce epipolar lines that go through the actual matches of feature points. In conclusion, we can state that, based on experiment, the best approach for the stereo setup calibration is to compute the projection matrices independently from feature points whose 3D locations are known. For the sake of consistency, the fundamental matrix used for matching can be computed by decomposition of the projection matrices, and the eight-point algorithm need not be invoked.

The eight-point algorithm minimizes quantities on the image plane, allowing one to get a fair guide for matching without any knowledge of the 3D geometry. In our case, estimation and decomposition of the projection matrices gives us the 3D geometry of the setup, as long as the positioning of the calibration pattern can be

made with sufficient precision. The experimental results show that it is the case. We can therefore use safely the information we have to build the fundamental matrix from (3.34) and get a self-consistent model of our stereo setup.

Chapter 4

Evaluation of the Camera Positions through 3D Reconstruction of Tracked Points

A calibrated stereo setup allows one to keep track of the motions in 3D space that a rigid object is experiencing in front of the cameras. Alternatively, a stereo setup mounted on a vehicle or a robot can compute its displacement, assuming a fixed rigid environment (or, at least, the fixed rigid environment represents a significant fraction of the fields of views of the cameras). In order to achieve such a task, matches must be identified between both images at a given starting frame. For each camera, the feature points corresponding to the matches must be tracked until the next frame. Reconstruction of the 3D points at the instant of the starting frame and at the instant of the next frame can be performed, and the two clouds of 3D points registered, as explained previously. Outliers must be identified, as they corrupt significantly the 3D registration results. This can be accomplished through a RANSAC algorithm, as will be explained in the next sections, and demonstrated experimentally.

The problem of camera pose estimation from a stereo setup has been addressed by various researchers, through different paths. In [22], a binocular or trinocular stereoscopic setup is used and its path along a sequence is computed by using tridimensional

reconstruction and registration. The robustness to matching and tracking errors is provided by two means. First, trilinear tensors are computed between image triplets. The features that support the trilinear tensors are known to be reliable. Second, a random sample consensus (RANSAC) [8] algorithm is applied to the 3D registration procedure. It is assumed that the disparities of the tracked feature points across the whole sequence is less than one third of the image size, thus constraining the camera movements.

In [23], the goal is to compute the registration of two consecutive scene captures along with the extrinsic calibration parameters of the stereo setup and the 3D location of a minimum of four matched and tracked feature points. The essential matrix of the stereo setup is calculated from the eight correspondences given by the four feature points in both captures, and nonlinear methods are used to enforce its constraints. The essential matrix is decomposed to retrieve the extrinsic calibration parameters up to a scale factor of the translation vector. At this point, 3D reconstruction can be applied to the feature points, yielding two clouds of a minimum of four 3D points. The registration between the two captures can then be calculated. This approach differs from the proposed method in the fact that they do not compute the extrinsic calibration parameters of the stereo setup prior to the computation of the registration. As a consequence, the matching process cannot be guided by the epipolar constraint. No experimental results along a sequence were shown to display the accumulation of error.

In [24], stereoscopic vision and shape-from-motion are combined in an attempt to exploit the strengths of both approaches, i.e. accurate 3D reconstruction for stereo and easy feature tracking for visual motion. The authors compute 3D reconstruction of feature points and the camera motion in two separate steps. They limited their experimentations to short sequences where the viewpoints don't change dramatically from the first to the last capture.

The interested reader can find additional research topics on motion of a stereoscopic setup in the literature ([37], [38], [39]), [40], [41]).

4.1 Matching

In the scope of this thesis, the word "matching" will be used to designate the task of finding matches between the left and the right camera at a given instant. We will assume that the fundamental matrix is available, obtained through decomposition of the projection matrices. Furthermore, it will be assumed that the cameras have approximately the same orientation, and have moderately small baselines. These assumptions are required to facilitate the matching. Under these assumptions, when comparing small windows around a given pair of pixels, these pixels will be acknowledged as a match if their surrounding windows look alike. If these assumptions are not satisfied, it would be a case of wide-baseline matching, which is a significant challenge in itself (cf. [25], [26], [27], [28], [29], [30]). Since it is not the core of this thesis, it won't be addressed.

Vincent and Laganière presented a comparative study of the narrow-baseline matching strategies in [31]. Although they assumed an uncalibrated image pair, we can adapt the general matching procedure they used:

1. Identify corners in both images;
2. Identify, for a given corner in image 1, the corners in image 2 that are close enough to the corresponding epipolar line;
3. Compute the level of dissimilarity between the corner in image 1 to be matched with each of the candidate corners in image 2.

The first step accomplished by the implemented matching function is to identify the corners in both images. This is performed by the function cvPreCornerDetect(), which is part of the Intel OpenCV package. This function itself calls a function which performs an edge detection operation twice (once in the x-direction, once in the y-direction) through a convolution with a 3×3 Sobel kernel. After this operation, each pixel has a value associated with its corner strength. A threshold is passed as a parameter that allows to filter out the pixels whose corner strength is too low.

The second step is to identify, for a given corner in image 1, the corners in image 2 that are close enough to the corresponding epipolar line, obtained from the fundamental matrix. The maximum distance of a candidate corner to the epipolar line is passed as a parameter. Experiments showed that a maximum distance of 1 pixel is a good choice in the vast majority of the cases (c.f. Section 3.3.4).

The third step is to compute the level of dissimilarity between the corner in image 1 to be matched with each of the candidate corners in image 2, that are close enough to the epipolar line. This is achieved by defining a square window around the corner of interest in image 1, and comparing with the surrounding windows of each candidate corner in image 2. The two windows to be compared are removed of their respective average value (in order to compensate somehow for illumination differences), then subtracted, and the obtained error is the sum of the squares of the entries of the difference matrix, divided by the number of pixels in the window. This is the dissimilarity measurement.

$$dissimilarity = \frac{\sum_{window}[(GL_{left} - \mu_{left}) - (GL_{right} - \mu_{right})]^2}{N_{window}} \tag{4.1}$$

where GL stands for *gray level*, μ is the average gray level over a given window and N_{window} is the number of pixels in the window.

The best match is the candidate corner that has the lowest dissimilarity level. The maximum dissimilarity level is passed as a parameter, that determines whether the best match for a given corner of interest in image 1 is good enough to be kept as part of the returned list of matches.

The values of the three parameters (the corner strength threshold, the maximum distance from the epipolar line and the maximum dissimilarity) must be determined manually, based on the sharpness of the images, the desired number of matches that must be found and the available processing time.

Figure 4.1 shows a pair of stereo images with the matched points found by the algorithm just described. For those interested in research topics on stereo matching, there is a vast literature available (cf., for instance, [32], [33], [34], [35])

Figure 4.1: Stereo pair with matched points

4.2 Tracking

The tracking function takes two images taken by the same camera at different instants, and a list of feature points to be tracked from instant 1 to instant 2. Instead of searching in a linear area defined by an epipolar line and a tolerance distance as in the case of the matching function, the tracking function must search in a disk whose center and radius are parameters. In general, the number of candidate matches is a lot larger than in the case of the matching function, because the area of search is larger. The mechanisms of identifying corners and measuring dissimilarity through windows is the same as described in Section 4.1, although the values of the optimal parameters may differ for tracking and matching, when applied to a given sequence.

Figure 4.2 shows the result of the tracking algorithm, with corresponding feature points that were tracked.

4.3 Robust Registration Algorithm

After having found matches and tracked the points in both sequences, two clouds of $3D$ points can be constructed, based on the matches at instant 1 and their tracked correspondents at instant 2. These two clouds of $3D$ points can be registered to find

Figure 4.2: Pair of consecutive captures with tracked feature points

the rigid motion of the object, as described in Section A.3. Unfortunately, direct use of the raw data is not appropriate, since the false matches and the tracking errors will corrupt the result. Instead, it is necessary to incorporate a random sample consensus (RANSAC) algorithm that will filter out the bad pairs of 3D points.

A minimum of 3 pairs of non-collinear 3D points are necessary to perform a 3D registration. As a consequence, the first step of the algorithm will consist of finding a trio of pairs of 3D points that do not constitute a degenerate case.

4.3.1 Random Drawing of a Trio of 3D Matches

In order to make sure that a randomly drawn trio of 3D matches are not in a collinear configuration neither clustered together, two conditions must be imposed:

- The distance between any two points in the trio must be greater than a given minimum. This condition excludes tight clusters of points;
- The area defined by the three points must be greater than a given minimum. This condition excludes collinear configurations.

The first item alone is not sufficient since three collinear points that are located far apart would satisfy it, while the second item alone would allow a trio constituted of

two points close from each other with a third point far away, such that the area of the triangle is sufficient. A third condition, which is not related to collinearity of the points, must be added before accepting a trio of 3D matches as a valid candidate. The trio must agree itself with its corresponding registration, i.e. when applying $\vec{X}|_{Ref1} = R_{reg}\vec{X}|_{Ref2} + \vec{T}_{reg}$, the error between a 3D point and the transformation of its counterpart must be less than a given distance. This condition ensures that the 3D matches agree with each other, and therefore excludes any obvious outlier from the candidate trio.

4.3.2 Count of the Number of Matches that Agree with the Candidate Registration

Given a candidate registration, as the transformation linking the two trios of 3D points, a count of the number of agreeing pairs can be performed (the support set). For each pair of 3D points, if the distance between $\vec{X}|_{Ref1}$ and $R_{reg}\vec{X}|_{Ref2} + \vec{T}_{reg}$ is less than a maximum distance (a parameter), then this pair is said to agree with the candidate registration. The whole procedure of Sections 4.3.1 and 4.3.2 is repeated several times. The number of trials can be set such that the probability of success at finding at least one trio of good matches is above a desired value (cf. [8]). The candidate registration having the highest number of agreeing matches is declared the best candidate registration.

The maximum distance parameter chosen was 2 mm. It is significantly higher that the reconstruction error displayed on Figure 2.7. The experiment of Figure 2.7 was performed using sub-pixel accuracy with a calibration pattern, which is not achievable on a real-world object.

4.3.3 Identification of the Good Matches and Final Registration

Finally, all the matches that agreed with the best candidate registration are used to compute the final output registration. The list of the good 3D pairs can also be supplied. The size of this list is a good indication of the quality of the registration.

4.3.4 Summary of the Algorithm

1. Do

   (a) Randomly draw a candidate trio of 3D matches that is not in a degenerate configuration

   (b) Compute R_{reg} and $\vec{T}_{reg}$ with the candidate trio, through the algorithm described in Section A.3.

   (c) Count the number of matches that agree with the candidate registration

   while the probability of success at finding at least one trio of good matches is below a desired value

2. Identify all trios that agree with the candidate registration having the highest score and perform the final registration with these trios

4.4 Computation of the New Camera Positions

From the registration homogeneous transformation Q_{reg}, the new positions of the cameras can be computed, from (A.79). The projection matrices of the cameras at the new positions can therefore be built from (2.11). It is not necessary to match and perform 3D reconstruction at every capture: one could simply track feature points and match at a lower frequency (1 capture out of 5, for instance).

One of the main problems associated with such a technique is the accumulation of error, since every new position is computed from the previous. It is assumed that no

special target points that could allow recalibration are available on the object or in the environment. Instead, one must rely on the knowledge of the approximate camera positions to identify points of view that were previously captured (loop detection). This information will be used to correct for the drift, each time the cameras pass by a location where they have been before.

4.5 Detection of Previously Viewed Locations

The goal of this procedure is to identify, in a sequence, camera positions that are close to their previous positions in an earlier image capture. Whenever such a loop is detected, tracking is possible between the earlier and the later pair of views. This allows the registration between the two positions and correction of the accumulated error.

As pointed out in Section 4.1, we won't address the situations of wide-baseline matching or tracking. This means that, in order for the described tracking scheme to work, two conditions must be met:

- The Z-axes of the two views must be nearly parallel;
- The distance between the center of projection of the views must be sufficiently small.

At first sight, it is not sufficient that the Z-axes be nearly parallel, it is also necessary that the Y- (or the X-) axes be nearly parallel for the correlation technique to work. Nevertheless, we can relax this constraint since our knowledge of the approximate camera positions will allow us to de-rotate the images around their Z-axes in such a way that they are sufficiently aligned.

The distance between the center of projection of the views is directly calculated from the length of the vector going from one center to the other. In order to calculate the maximum distance we can afford, we must take into consideration the fact that the two views may be collinear along their parallel Z-axes (i.e. one view may be

in front of the other), resulting in a scale difference between the two images. The closer the object of interest will be to the cameras, the smaller the tolerance on the distance between the views will be, since the tracking algorithm is obviously not scale invariant.

The angle between the Z-axes of two views can be computed through a scalar product of unit vectors parallel to the Z-axes of the two cameras, as expressed in the world reference frame:

$$\hat{k}_M|_W = Q_{C_M/W} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.2}$$

$$\hat{k}_N|_W = Q_{C_N/W} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.3}$$

$$\cos(\theta) = \hat{k}_M|_W \cdot \hat{k}_N|_W \tag{4.4}$$

where $Q_{C_M/W}$ and $Q_{C_N/W}$ are the homogeneous transformation matrices linking a camera at Capture M and at Capture N with respect to the world reference frame.

The angle between the Z-axes of the left camera at capture M and N need not be the same as the equivalent for the right camera. In a sequence, the minimal angle (or distance) with respect to a given frame may not happen at the same frame for the left and the right camera. When trying to identify the best capture to be matched with an earlier capture, we must find a compromise between the two cameras.

Whenever a view is detected as being close to a previously captured view, the drift of the later view can be compensated. Of course, it is assumed that the earlier the view, the better the accuracy, since its location has been computed from a smaller number of cascaded transformations.

4.6 Tracking between Non-Consecutive Captures

4.6.1 Identification of the Rotation Angle Around the Z-Axis

As discussed previously, a pair of views are similar if their Z-axes are nearly parallel, but they can have a wide angular difference around their Z-axes. Since the tracking algorithm is not rotation invariant, this situation could prevent the identification of correspondences. We can overcome this difficulty by making use of the knowledge we have of the approximate positions of the camera. We will be searching for the rotation that must be applied to the images of the later view, such that it is as aligned as possible with the earlier view.

Aligning the Y-Axes

In a first approach, we will aim at minimizing the angle between the Y-axes of two views by applying a rotation around the Z-axis of the second view. Let us state the result:

Let r_{ij} be the element (i, j) of the rotation matrix linking the view 2 with the view 1, $R_{C2/C1}$.
If $r_{10}sin(\arctan(-\frac{r_{10}}{r_{11}})) < r_{11}cos(\arctan(-\frac{r_{10}}{r_{11}}))$, then:

$$\alpha_Y = \arctan(-\frac{r_{10}}{r_{11}}) \tag{4.5}$$

else:

$$\alpha_Y = \arctan(-\frac{r_{10}}{r_{11}}) + \pi \tag{4.6}$$

Proof. The rotation component of a reference system built with a pure rotation α around the Z-axis of the second reference system is:

$$R_{C2/C1}R_{(\alpha,0,0)} = \begin{bmatrix} r_{00}\cos(\alpha) + r_{01}\sin(\alpha) & -r_{00}\sin(\alpha) + r_{01}\cos(\alpha) & r_{02} \\ r_{10}\cos(\alpha) + r_{11}\sin(\alpha) & -r_{10}\sin(\alpha) + r_{11}\cos(\alpha) & r_{12} \\ r_{20}\cos(\alpha) + r_{21}\sin(\alpha) & -r_{20}\sin(\alpha) + r_{21}\cos(\alpha) & r_{22} \end{bmatrix} \tag{4.7}$$

A unit vector, oriented along the Y-axis of the camera 2 will be expressed, in the reference frame of camera 1:

$$\hat{j}_2|_{C1} = R_{C2/C1} R_{(\alpha,0,0)} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.8}$$

$$\overset{(4.7)}{=} \begin{bmatrix} -r_{00}\sin(\alpha) + r_{01}\cos(\alpha) \\ -r_{10}\sin(\alpha) + r_{11}\cos(\alpha) \\ -r_{20}\sin(\alpha) + r_{21}\cos(\alpha) \end{bmatrix} \tag{4.9}$$

We aim at maximizing the scalar product between the Y-axes of the two cameras:

$$\hat{j}_2|_{C1} \cdot \hat{j}_1|_{C1} = \hat{j}_2|_{C1} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.10}$$

$$= -r_{10}\sin(\alpha) + r_{11}\cos(\alpha) \tag{4.11}$$

To maximize the scalar product (4.11), we pose its first derivative with respect to α equal to 0 and we pose its second derivative negative:

$$\frac{\partial}{\partial\alpha}(\hat{j}_2|_{C1} \cdot \hat{j}_1|_{C1}) = \frac{\partial}{\partial\alpha}(-r_{10}\sin(\alpha) + r_{11}\cos(\alpha)) \tag{4.12}$$

$$= -r_{10}\cos(\alpha) - r_{11}\sin(\alpha) = 0 \tag{4.13}$$

$$\frac{\partial^2}{\partial\alpha^2}(\hat{j}_2|_{C1} \cdot \hat{j}_1|_{C1}) = \frac{\partial}{\partial\alpha}(-r_{10}\cos(\alpha) - r_{11}\sin(\alpha)) \tag{4.14}$$

$$= r_{10}\sin(\alpha) - r_{11}\cos(\alpha) < 0 \tag{4.15}$$

Together, constraints (4.13) and (4.15) yield to (4.5) and (4.6). □

Aligning the X-Axes

Alternatively, one can aim at minimizing the angle between the X-axes of the two cameras by applying a rotation α_X around the Z-axis of the later view. It can be shown that, in the case where the Z-axes are perfectly aligned, the two angles α_Y and α_X are equal (the two reference frames can be made to coincide). In the general

case where the Z-axes are not perfectly parallel, the optimal angles α_Y and α_X will not be equal. The optimal angle α_X that will minimize the angle between the X-axes is given by the following relations:

If $-r_{00}\cos(\arctan(\frac{r_{01}}{r_{00}})) < r_{01}\sin(\arctan(\frac{r_{01}}{r_{00}}))$, then:

$$\alpha_X = \arctan(\frac{r_{01}}{r_{00}}) \tag{4.16}$$

else:

$$\alpha_X = \arctan(\frac{r_{01}}{r_{00}}) + \pi \tag{4.17}$$

Proof. The scalar product of the unit vectors parallel to the X-axes of both cameras will be:

$$\hat{i}_2|_{C1} \cdot \hat{i}_1|_{C1} = (R_{C2/C1} R_{(\alpha,0,0)} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.18}$$

$$\overset{(4.7)}{=} r_{00}\cos(\alpha) + r_{01}\sin(\alpha) \tag{4.19}$$

Setting the first derivative of (4.19) with respect to α equal to 0 and its second derivative negative:

$$\frac{\partial}{\partial\alpha}(\hat{i}_2|_{C1} \cdot \hat{i}_1|_{C1}) = \frac{\partial}{\partial\alpha}(r_{00}\cos(\alpha) + r_{01}\sin(\alpha)) \tag{4.20}$$

$$= -r_{00}\sin(\alpha) + r_{01}\cos(\alpha) = 0 \tag{4.21}$$

$$\frac{\partial^2}{\partial\alpha^2}(\hat{i}_2|_{C1} \cdot \hat{i}_1|_{C1}) = \frac{\partial}{\partial\alpha}(-r_{00}\sin(\alpha) + r_{01}\cos(\alpha)) \tag{4.22}$$

$$= -r_{00}\cos(\alpha) - r_{01}\sin(\alpha) < 0 \tag{4.23}$$

Together, constraints (4.21) and (4.23) yield to (4.16) and (4.17). □

Since there is no *a priori* reason to believe that it is more important to align the X-axes nor the Y-axes, we will use a rotation angle that is the average value of α_X and α_Y. The center of the rotation that must be applied to the later images is

the principal point of the camera, as it is defined as the location where the Z-axis intersects the image plane.

4.6.2 Estimation of the Location of the Feature Points to Track

Theoretically, we have everything we need to apply the tracking algorithm described in Section 4.2: From an earlier pair of images, matches are extracted, that will constitute feature points to track. These feature points can be searched for in the rotated later images, such that a correlation technique can be applied. The 3D reconstructions of the feature points in the earlier pair and their tracked correspondences (de-rotated, so they correspond to pixel values of the non-rotated cameras) give two clouds of 3D points that can be used to compute the new locations of the cameras at later time N. In practice, the tracking algorithm can be enhanced significantly, because we can predict where, in the rotated images, the feature points should be found. In the basic tracking algorithm where we track feature points in a sequence of images, we assume the feature points at time $M + 1$ will lie close to their location at time M, so the search area will be a disk of a given radius around its location at time M. In the case we are dealing with, such an assumption is not justified, but we still can predict where a given feature point should be, by using our approximate knowledge of the projection matrices. We know the 3D location of all the feature points, so all we have to do is to project these 3D locations using the approximate projection matrices of both cameras to obtain a good estimate of the location of the feature points at later time N. Of course, these locations must be rotated around the principal point, to correspond to pixel values of the rotated images.

4.7 Accumulated Error Correction

Let us assume the view N has been identified as being close to the earlier view M, and corrected accordingly. We computed a correction matrix that can be premultiplied to the initially computed location of the view N.

$$Q_{N(corrected)} \equiv Q_{correction,N} Q_{N(initial)} \tag{4.24}$$

The problem we would like to address now is how to correct the intermediary views, which cannot be tracked from previous views other than their immediate neighbors. We assume we now have a high level of confidence in the knowledge of the location of the view M and the view N, $N > M$. How can we correct the views $M+1, M+2, ..., N-1$?

4.7.1 Linear Interpolation

The most naive approach would be to consider the correction matrix parameters $\{\theta, \phi, \psi, Tx, Ty, Tz\}$ as independent, and linearly interpolate them through

$$p_n = p_N \times \frac{n-M}{N-M} \tag{4.25}$$

where p_n is the parameter of interest $\{\theta, \phi, \psi, Tx, Ty, Tz\}$ of the correction matrix of view n. This approach will be referred to as the linear interpolation method.

4.7.2 Uniform Distribution of The Correction

Let us assume the drift in the calculated location of the views was uniformly distributed over all the registration steps. Furthermore, let us assume the individual registration steps along with the error in the registration had small rotation components. Let us model the uniform error in the following way, rewriting (A.79) with the introduction of Q_{error}, the unit error transformation matrix that happened at every registration:

$$Q_{n(calculated)} = Q_{error} Q_{reg,n-1} Q_{n-1(calculated)} \tag{4.26}$$

Through a cascade of these matrices:

$$Q_{n(calculated)} = \left(\prod_{i=n-1}^{M} Q_{error} Q_{reg,i} \right) \times Q_{C_M/World} \tag{4.27}$$

Under the assumption of small rotation components of $\{Q_{reg,i}\}$ and Q_{error}, we can commute matrices in such a way that we gather the error matrices to the left and take them out of the product, using (B.27)

$$Q_{n(calculated)} \overset{(B.27)}{\simeq} Q_{error}^{n-M} \left(\prod_{i=n-1}^{M} Q_{reg,i} \right) \times Q_{C_M/World} \tag{4.28}$$

The correction matrix is assumed to annihilate the error of the view N. Therefore

$$Q_{correction,N} = (Q_{error}^{N-M})^{-1} = Q_{error}^{M-N} \tag{4.29}$$

$$Q_{error} = Q_{correction,N}^{\frac{1}{M-N}} \tag{4.30}$$

The correction matrix at view n will have to annihilate the error at view n, i.e.

$$Q_{correction,n} = (Q_{error}^{n-M})^{-1} = Q_{error}^{M-n} \tag{4.31}$$

$$\overset{(4.30)}{=} Q_{correction,N}^{\frac{M-n}{M-N}} \tag{4.32}$$

$$= Q_{correction,N}^{\frac{n-M}{N-M}} \tag{4.33}$$

Equation (4.33) gives the correction matrix that must be premultiplied to the calculated location of a camera at view n, given the correction matrix at view N, under the assumption of uniform distribution of the error along the registration steps, and under the assumption of small rotation components of both the registration matrices and the error transformation matrix.

Both the linear interpolation technique and the uniform distribution of the correction can be extended to the case of non-uniform distribution of the error. One can

have some information about which registration steps contributed most to the overall error, according to some *suspicion level*. In this scheme, the ratio $\frac{n-M}{N-M}$ in equations (4.25) and (4.33) would be replaced by some factor α. This factor would increase smoothly between 0 (for view M) and 1 (for view N) according to the suspicion level of each registration step. For instance, the number of 3D pairs of points that were used in the robust registration procedure could be used as a measure of the suspicion level (a high number of 3D pairs corresponds to a low suspicion level). In all cases, the correction matrix takes care of the average component of the error. If the error of the intermediary views had an oscillating behavior around the average component, there is no way to correct it.

4.8 Experimental Results

The movement of a Russian headstock was recorded by a stereo setup. Figure 4.3 shows the sequence, as recorded by the right camera.

The number of initial matches and the number of good 3D pairs are plotted in Figure 4.4. The number of initial matches lie in the range of a few thousands and the number of 3D pairs used for registration lie in the range of a few tens to a thousand. The filtering of the RANSAC algorithm is severe. This can be explained by the conditions that are necessary for a given initial match to be used by the RANSAC algorithm: it must be a good match correctly tracked in both the left and the right image.

The projection matrices of the cameras at each position have been computed by matching, tracking and 3D registration of reconstructed points. The estimated angles between the Z-axes and distances between the centers of projection, with respect to the first capture, have been plotted in Figure 4.5. The minimum angle and distance happens at Capture 19 for the left camera. For the right camera, the minimum distance happens at Capture 15, while the minimum angle happens at Capture 16. For the sake of the illustration, we will correct the error at Capture 18, but it probably

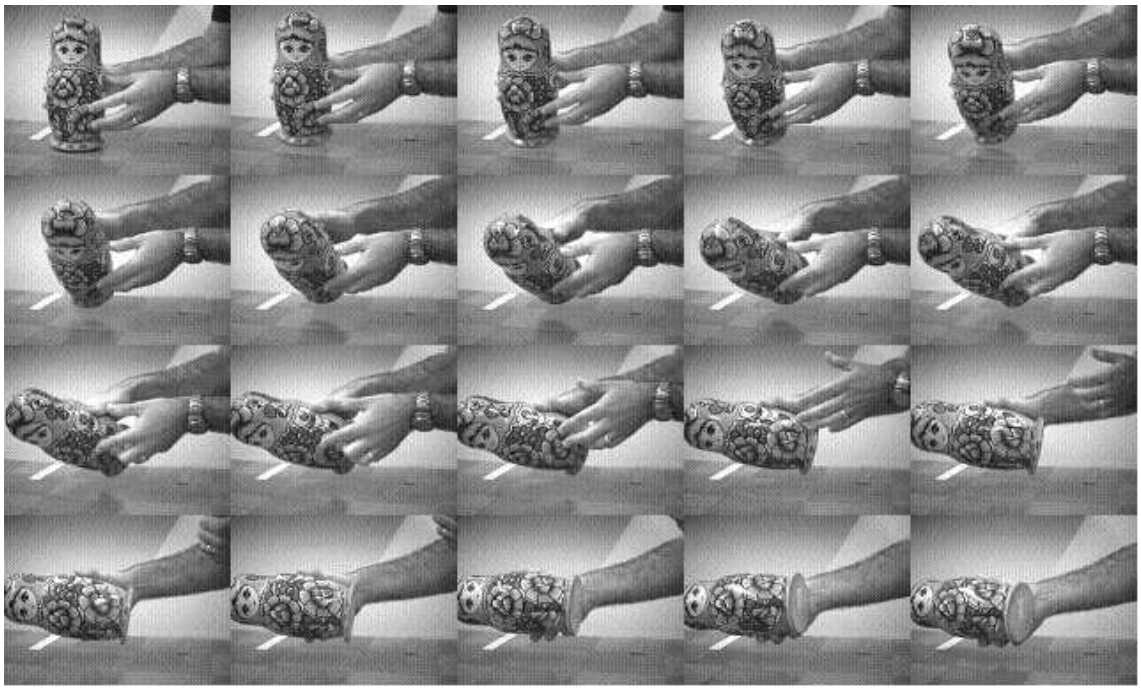


Figure 4.3: Russian headstock sequence recorded by the right camera

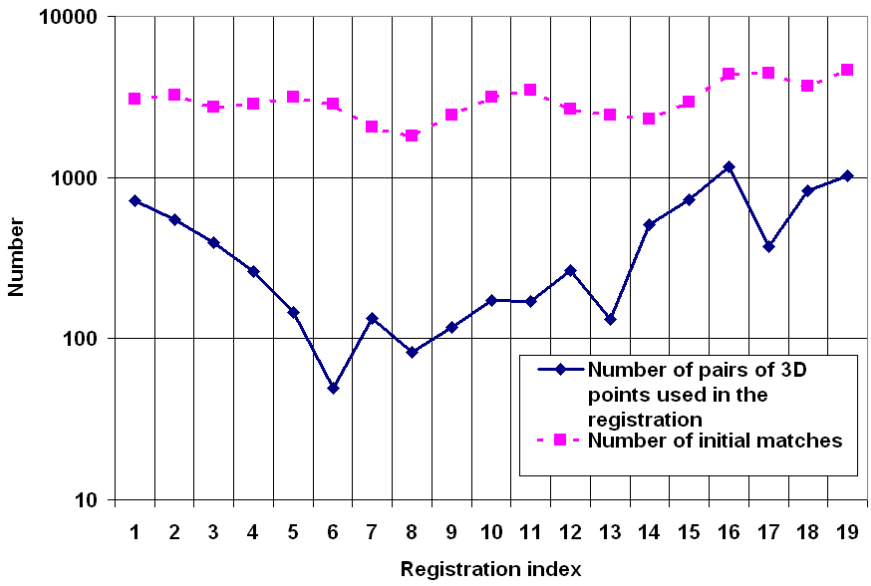
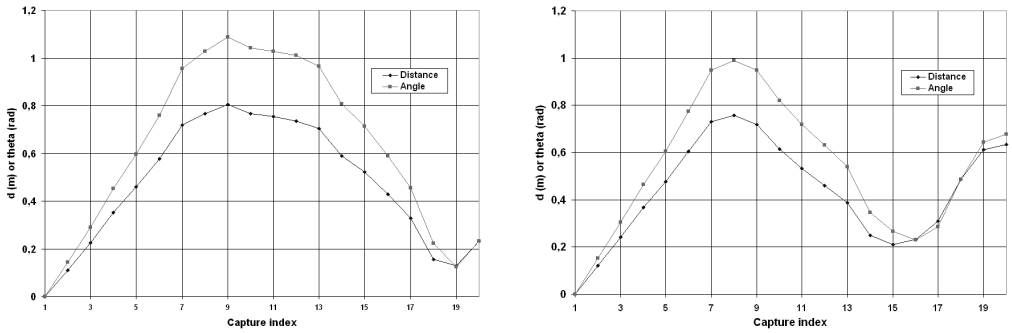


Figure 4.4: Number of initial matches and 3D pairs used for registration of the Russian Headstock sequence

could have been done for Captures 16 to 19.



(a) (b)

Figure 4.5: (a) Angle and distance of the left camera, with respect to the first capture (b) Angle and distance of the right camera, with respect to the first capture

The rotation angles around the Z-axes of the left and right cameras were estimated to be -1.54 rad and -1.49 rad respectively. The corresponding rectified images are shown in Figure 4.6.

Matches were identified in the two images of the first capture, and tracked in their correspondent image in Capture 18, searching in the vicinity of where the features are expected to be found, according to the approximate projection matrices available. The corrected projection matrices perform better than the initial projection matrices, obtained by a cumulation of transformations. This can be seen from Figure 4.7, where the 3D points obtained from the robust registration procedure between Capture 1 and Capture 2 (in other words, good 3D points) have been projected in the left image, using the uncorrected and the corrected projection matrices, respectively. It can be observed, especially on the circumferences of the face, the center flower and the right flower, that the projections fall precisely on the features in the case of the corrected matrix, while they fall a little below their expected location, in the case of the uncorrected matrix. The error is not huge. In the worst case, it is around 15 pixels, which means that the error accumulation is low. After 17 registrations, the feature points observed on the first pair of images can still be found in a radius of 15 pixels around their expected locations.

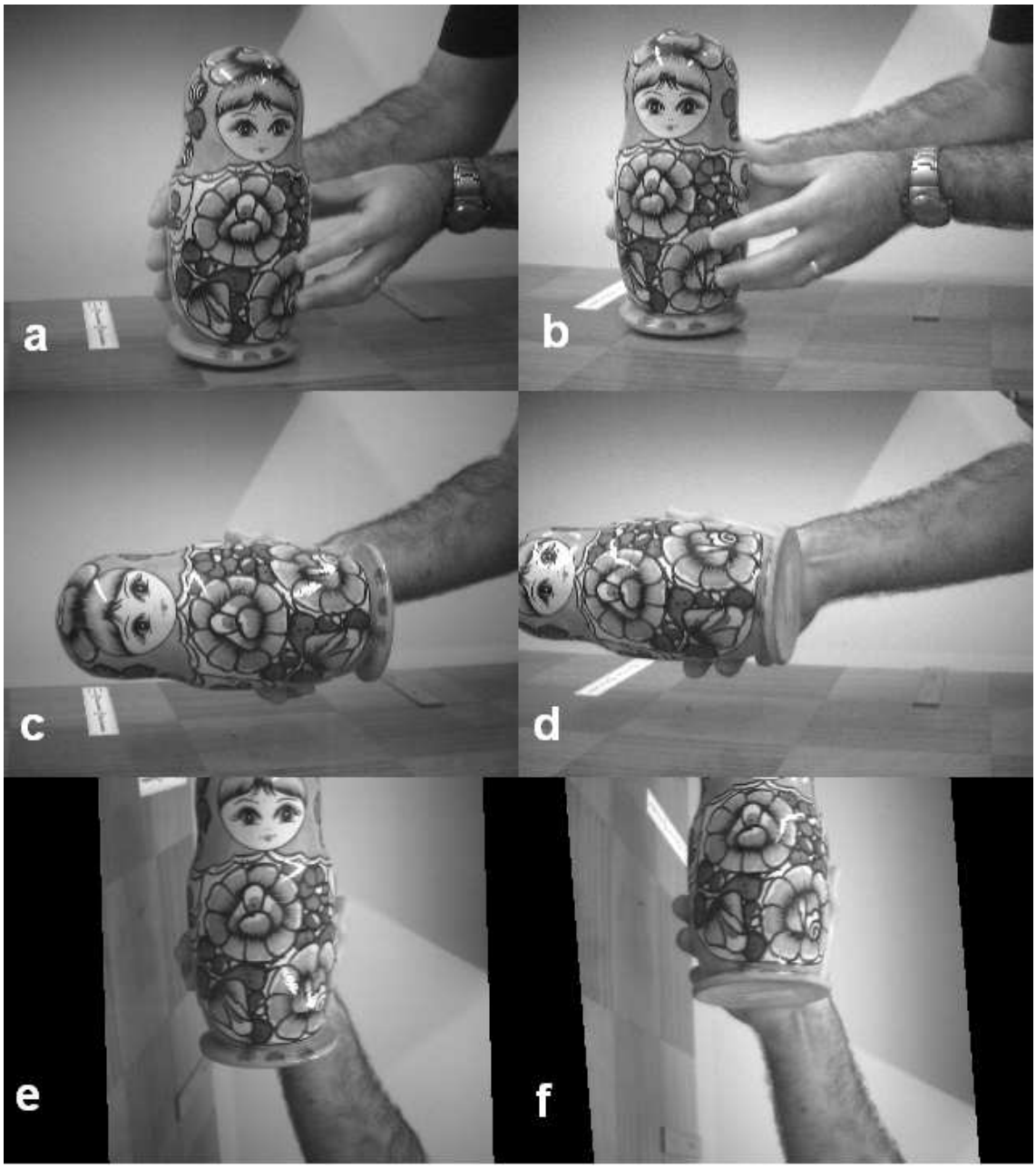


Figure 4.6: (a) Initial left image; (b) Initial right image; (c) Left image after 17 registrations; (d) Right image after 17 registrations; (e) Optimally rotated left image; (f) Optimally rotated right image. These two transformed images can now be matched with images (a) and (b). The top and the bottom of the images were discarded since the tracking function takes only images of the same format.

(a)

(b)

Figure 4.7: (a) Projection of original 3D points in the left image (capture 18) using the uncorrected projection matrix. In order to emphasize the reprojection error, lines have been drawn between reprojected and expected locations, for some points. (b) Projection of original 3D points in the left image (capture 18) using the corrected projection matrix

Chapter 5

Comparison between 3D Reconstruction Methods

At the core of the technique described in chapter 4 lies the 3D reconstruction; It is a fundamental tool in stereoscopic vision applications. The task of 3D reconstruction of a feature point can be carried out in a number of ways. All the experimental results generated thus far were obtained by least-squares solving from the projection matrices, as described in section 2.5.2. This chapter will provide an experimental justification for doing so. We will also assess the validity of the error correction scheme proposed in section 4.7, through an experimental comparison with a bundle adjustment commercial software.

5.1 Comparative Results between Least-Square Solving from the Projection Matrices and Triangulation

Starting with the same calibration information, least-squares solving from the projection matrices and the triangulation method (c.f. section 2.5.1) are two competing techniques. The first one aims at minimizing an algebraic error quantity while the

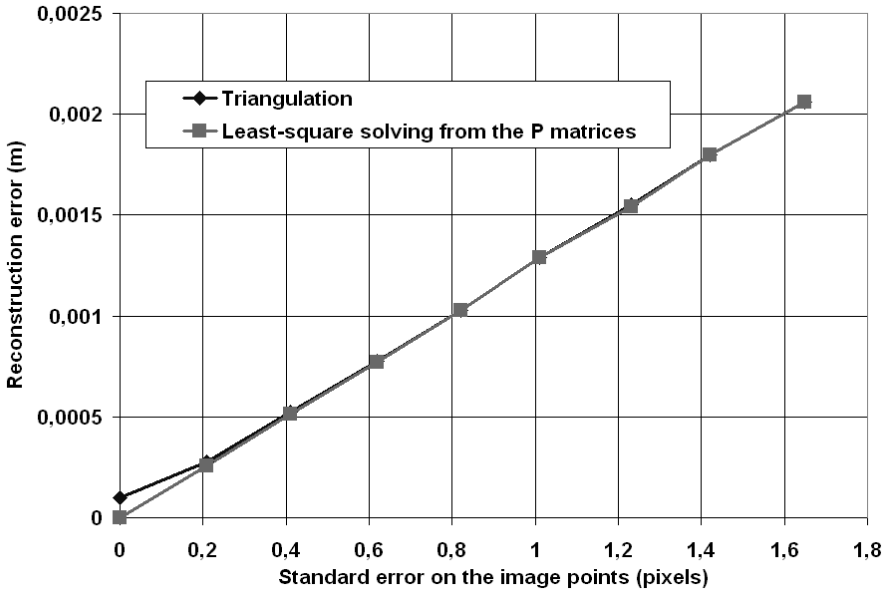


Figure 5.1: Comparison between the triangulation method and least- squares solving from the projection matrices, with synthetic data

second one aims at minimizing the reconstruction error in the euclidian space.

5.1.1 Synthetic Data

One thousand random 3D points were generated in a volume of 30 cm × 30 cm × 30 cm. A virtual stereo setup typical of those presented in section 2.6 was created. Uncorrelated uniform random noise was progressively added to the image points and the standard reconstruction error was measured. As can be seen on figure 5.1, as soon as the standard error on the image points get higher than 0.2 pixels, the two curves are undistinguishable. In most situations, feature points will be rounded to the nearest pixel, which itself induces a standard error of roughly 0.4 pixels (the standard deviation from the exact 2D location, when uniform noise between -0.5 and +0.5 pixel is added, c.f. (B.29)). Therefore, the two reconstruction techniques are equally tolerant to errors in image points.

5.1.2 Measured Data

To evaluate the performance of the two methods with real data, we measured their ability to retrieve 3D points whose locations are known, starting with the same calibration information. We carefully positioned a calibration pattern at given locations with respect to a global reference frame. The volume over which the points were distributed was 12 cm $\times$ 9 cm $\times$ 27 cm. The stereo setup configuration was typical of those presented in section 2.6. The standard reconstruction error, in this case, is the standard deviation of the difference between the computed 3D location and its measured location. Of course, this is not an absolute accuracy since the measured location is itself subject to an error that we estimate to be well under one millimeter. Nevertheless, as can be seen in figure 5.2, least-squares solving from the projection matrices gives, consistently, slightly lower reconstruction error. This is assumed to be due to the additional numerical manipulations required prior to the triangulation reconstruction algorithm, to decompose the projection matrices. Since the chosen calibration procedure provides the projection matrices, it seems natural, and now it is confirmed experimentally, that the best reconstruction technique is least-square solving from the projection matrices.

5.2 Bundle Adjustment

Bundle adjustment is an iterative method of computing the camera pose and the 3D location of feature points, given a set of matches from different cameras and the intrinsic calibration parameters of the cameras. Strictly speaking, it is more than a 3D reconstruction technique: it performs, at the same time, extrinsic calibration of cameras and reconstruction of feature points. The initial estimates of the camera positions can be obtained through essential matrix decomposition (after computing the fundamental matrix or trifocal tensors) [18]. The problem is to minimize the sum of the euclidian squared distance between the reprojection of the 3D points and their corresponding image points ([19], [36]), by varying the cameras and the 3D points

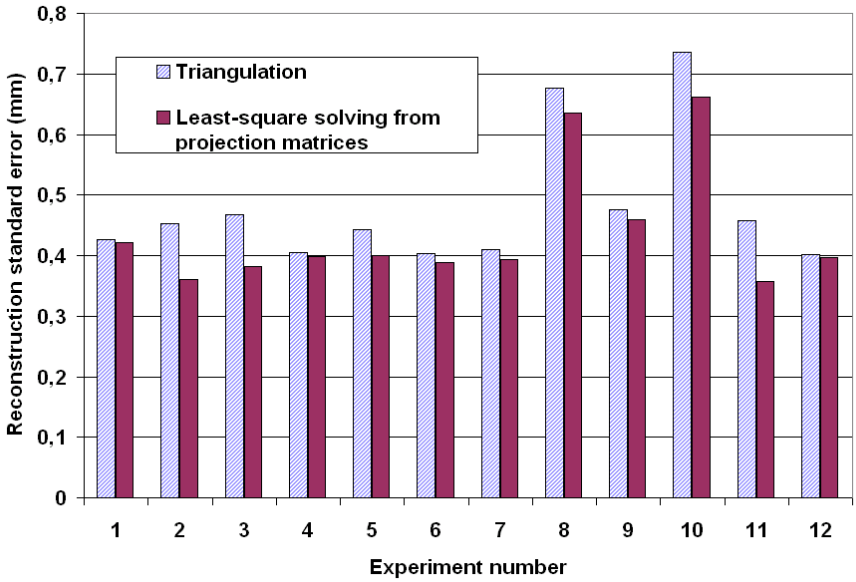


Figure 5.2: Comparison between the triangulation method and least- squares solving from the projection matrices, with real data

locations:

$$min_{P,Q}[\sum_i \sum_j d^2(\vec{u}(observed)_{i,j}, \vec{u}(reprojected)_{i,j}] \tag{5.1}$$

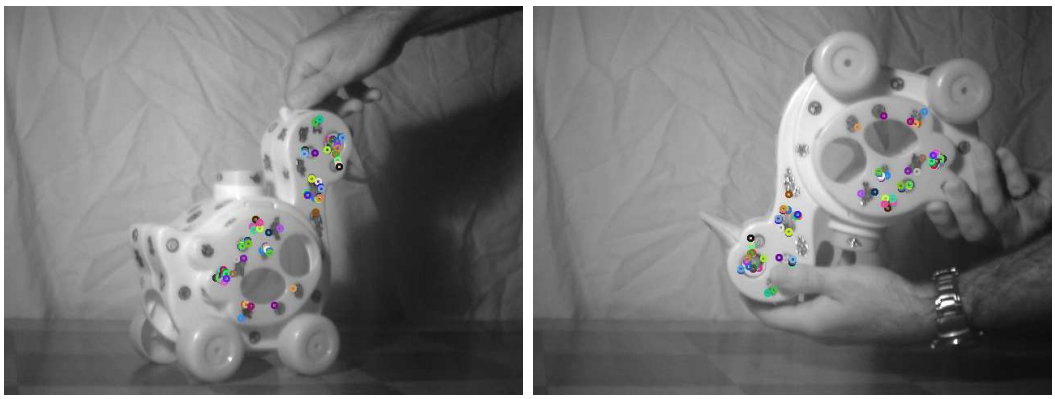
where P is the set of the extrinsic parameters of the cameras, Q is the set of the 3D points locations, $\vec{u}(observed)_{i,j}$ and $\vec{u}(reprojected)_{i,j}$ are the $i-th$ image point, observed and reprojected respectively, in the $j-th$ image and $d(\cdot,\cdot)$ is the euclidian distance.

When it converges correctly, bundle adjustment gives an optimal solution, i.e. the 3D configuration that best explains the observations in all views. For this reason, the bundle adjustment solution will be considered as the ground truth in order to validate our error correction scheme.

Bundle adjustment can hardly be automated, since it is very sensitive to false matches. Most commercial implementations of bundle adjustment rely on manual selection of matches, assuming they will all be good. PhotoModeler [1] is a commercial

[1]EOS Systems Technology (www.photomodeler.com)

software that implements bundle adjustment from a set of manually identified matches in a set of images. In order to compare the proposed method with bundle adjustment, PhotoModeler has been used to process a subset of the *Duck* sequence. The whole sequence could not be used, since the manual selection of matches is extremely time-consuming. Fourteen images were manually matched with 68 feature points. The returned camera positions are displayed in Figure 5.4, along with the 3D location of the selected feature points. The camera path forms two loops: 360^o in the $X-Y$ plane and a half-turn under the duck. These loops allow for error correction between captures 1 and 40 (for captures 2 through 40) and between captures 17 and 64 (for captures 41 through 64). Figure 5.3 shows the left images of captures 17 and 64, along with the projection of the 3D points used to register these two positions. During the second error correction, captures 17 to 40 were left untouched. The error correction matrix computed between captures 17 and 64 has been uniformly distributed in the position of captures 41 through 64.



(a) (b)

Figure 5.3: (a) Left image at capture 17, with the projection of the 3D points used for registration (b) Left image at capture 64, with the projection of the 3D points used for registration

Figure 5.5 shows the disagreement (in the position and the orientation of the cameras) between PhotoModeler and the proposed method, without error correction. The position disagreement is the distance between the computed centers of projection for both methods. The orientation disagreement is the angle between the $Z-$axes

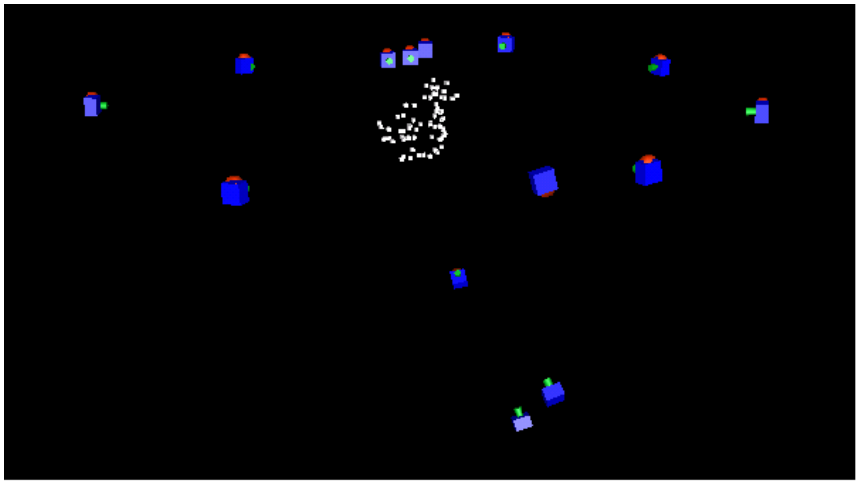
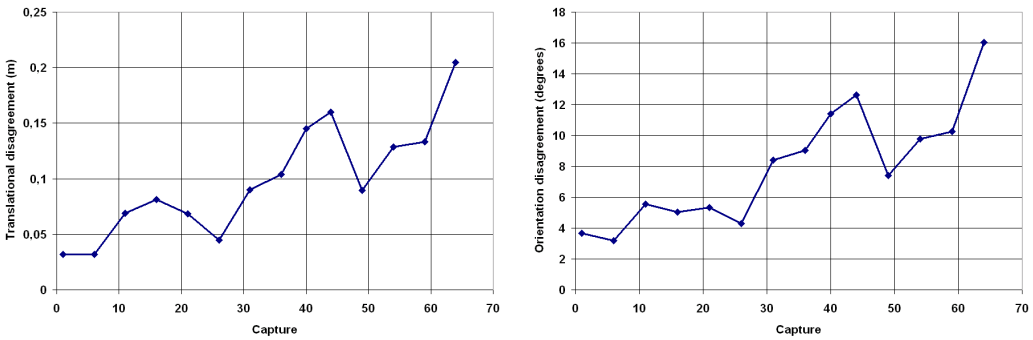


Figure 5.4: Positions of the left camera in the Duck sequence, as computed by PhotoModeler

of the camera reference frames, for both methods. As expected, the magnitude of the disagreement increases with the number of registrations, as the proposed method accumulates error. The PhotoModeler project had matches between the first and the last image, allowing for a closed loop configuration and thus preventing error accumulation.

It is worth noticing the fact that, as opposed to the proposed method, bundle adjustment does not grant any special status to the first capture. It can be adjusted, like every other camera position. In contrast, the proposed method gives a higher level of confidence in the earlier captures. The discrepancy between the two methods at the first capture is most probably related to errors in the bundle adjustment solution.

The bundle adjustment algorithm, in a first step, returns camera positions up to a scale factor and a global homogeneous transformation. To link it with absolute dimensions and a given reference frame, the user must provide the 3D locations of a triplet of feature points. As opposed to the calibration procedure described in section 3.1.1 where we used a large number of 3D points (a few hundreds), PhotoModeler uses only three 3D points to relate its arbitrary reference system to the one chosen by the user.



(a) (b)

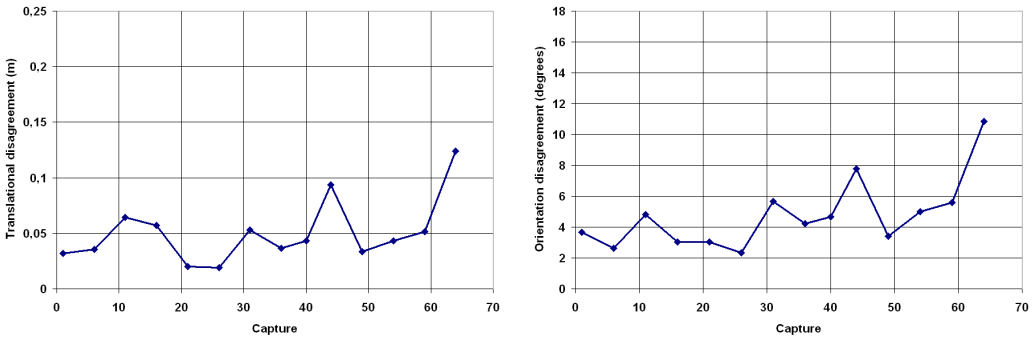
Figure 5.5: (a) Position disagreement between PhotoModeler and the proposed method, without error correction (b) Z-axis orientation disagreement between PhotoModeler and the proposed method, without error correction

Figure 5.6 shows the disagreement between PhotoModeler and the proposed method after the two passes of error correction through uniform distribution of the correction matrix. It can be seen that the disagreement magnitude increases a lot more slowly with the number of registrations, as compared with Figure 5.5, indicating that the error correction provided an improvement in the projection matrices.

Obviously, as the error increases with the number of registration, at a certain point, it won't be possible anymore to detect a loop since the feature points we'll be searching for will lie outside the searching area. Therefore, the error correction must be performed before the pose drift too much.

5.3 Summary

In this chapter, we performed experiments to identify the best 3D reconstruction method and to compare the proposed method with bundle adjustment. In particular, we investigated the validity of our error correction scheme, showing that it does provide an improvement on the calculated camera positions.



(a) (b)

Figure 5.6: (a) Position disagreement between PhotoModeler and the proposed method, after error correction (b) Z-axis orientation disagreement between PhotoModeler and the proposed method, after error correction

Chapter 6

Applications of the Camera Pose Estimation

6.1 Shape from Silhouette 3D Modelling

The action of building a volumetric representation of an object is referred to as 3D modelling. Virtual environments are populated by virtual objects, which are 3D models. They are building blocks in computer games, simulations and training applications. Furthermore, modelled heritage artifacts are extremely useful to archeologists, allowing them to virtually manipulate objects for which they cannot have a direct physical access.

In some instances, models must be built manually by a programmer, stitching together volumetric primitives. It is often the case in the video games industry, where most virtual objects are the offspring of an especially creative mind. In other instances, it is not the way to go. In heritage applications, the model has to be an accurate replica of the physical object, with all its imperfections. In medical training applications, it is reasonable to assume that it might be simpler to model automatically an existing physical model of a biological structure, rather than to manually reproduce its intricate shape.

Vision-based 3D modelling techniques can be subdivided in two broad categories:

- Active modelling;
- Passive modelling.

Active modelling techniques are those who send some form of probing energy towards the scene, under the form of a laser beam, a radar or a sonar. The two latter are used for modelling large scale environments, while the laser beam is most often used to build the model of a smaller artifact.

Passive modelling techniques use ambient light as the probing energy. One possible passive modelling technique would be to use a stereo setup, match every pixel of the images and perform 3D reconstruction on this large set of matches. A dense depth map would be obtained, and a mesh of triangular surfaces could be built from them. This approach would be computationally intensive, and the false matches could not be removed easily. Furthermore, complete volumetric reconstruction would require fusing the information from multiple depth maps, which would be difficult to achieve. Shape from silhouette is another, simpler instance of passive 3D modelling technique.

The basic idea behind shape from silhouette is the intersection of the silhouette cones. A silhouette cone is a volume defined by the silhouette of an object (i.e. the pixels identified as being different from the previously acquired background) projected through the center of projection of the camera (see Figure 6.1). A silhouette cone is completely determined by an image whose background has been extracted and the projection matrix of the camera. The silhouette cone provides a constraint on the volume occupancy of the object, since the object must be totally included inside it. When one has access to many silhouette cones, this upper limit on the volume occupancy gets tighter and tighter as the silhouette cones are intersected. In the limit, when an infinite number of silhouette cones from every possible view (outside the convex hull) are intersected, we get the *visual hull*. This concept has been introduced and analyzed by Laurentini ([11]). The visual hull is an upper limit to the volume occupancy of an object. An important feature to notice is the fact that if the object has hollows, such as the inside of a cup, the procedure described won't remove the voxels inside the hollows.

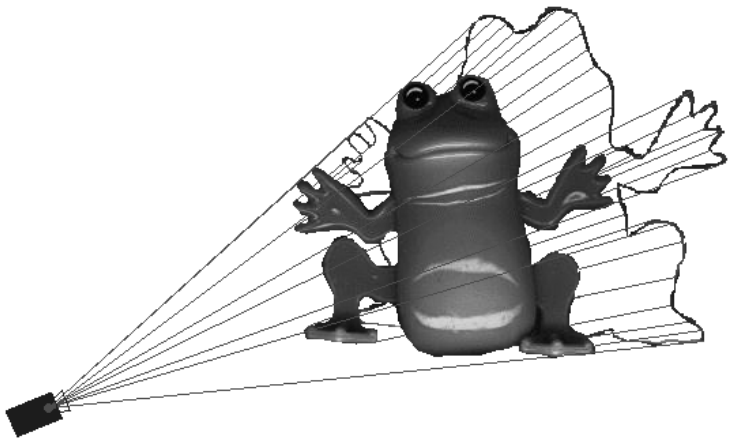


Figure 6.1: A silhouette cone

Shape from silhouette techniques have been the object of research for many years. Most of the papers in the literature make use of a turntable with step motors on which the object to be modelled is rotated in front of a camera. This feature facilitates the computation of the camera position, allowing the researchers to concentrate their efforts on voxel carving or model representation. In our case, our focus is on the computation of the camera positions, when they are allowed to move freely in space.

In [12], Kuzu and Rodehorst present a modelling system based on a single camera capturing images of an object on a turntable whose movement is known. Voxels are removed according to a voting scheme. As an additional feature, concavities are detected through a block-matching (correlation) algorithm.

In [13], Jones and Oakley introduce the *radial intersection set*, which is claimed to be a more compact representation of the visual hull than the octree, for an equivalent resolution. In this scheme, boundaries are represented by their intersection with radial lines, whose center is along the rotation axis of a turntable. The object to

be modelled is placed on the turntable such that the origin of the polar coordinates system is located inside the object.

In [14], Matusik et al. introduce a technique whose goal is to generate new views of an object in real time. Given a desired view and a reference (i.e. acquired) view, the epipolar geometry is computed. Then, every pixel of the desired view is scanned and an epipolar line is drawn in every reference view. Whenever any of the epipolar lines in the reference images lie outside the silhouette, the pixel of interest is declared as being part of the background. Otherwise, it is declared as being part of the object. This technique doesn't generate a list of opaque voxels and, as a consequence, is claimed to minimize artifacts resulting from volume quantization.

In [15], Snow, Viola and Zabih formulate the problem of labelling the voxels as an energy minimization problem. Their energy function includes a penalty term for assigning a given label to a given voxel and a penalty term that takes into account the labelling difference between a given voxel and its neighbors. This second term is a smoothness factor, which contributes to eliminate the hole drilling effect often observed when the background is not correctly extracted. This energy function is shown to be a member of a class of minimization problems that can be solved through a graph cut algorithm.

We used a simple voxel carving algorithm. Basically, we kept a given voxel label as "opaque" until its eight vertices were projected out of the silhouette, in one of the views:

- Input:
  - A list of silhouettes
  - A list of projection matrices corresponding to the silhouettes
  - A list of initially opaque voxels
- Output: A list of opaque voxels belonging to the object volume occupancy
- Algorithm:

- For each silhouette and its projection matrix
  - For each opaque voxel in the list
    - Project the eight voxel vertices in the silhouette image using the projection matrix
    - If the eight projections fall out of the object silhouette, then remove the voxel from the list
    - Otherwise, keep it

This algorithm is "careful": if any of the vertices is projected in the positive area of the silhouette, this voxel won't be removed. This approach is justified in this situation since false negative pixels have a stronger impact on the model than false positive pixels. False negative pixels (i.e. those labelled as background while they should be labelled as part of the object) will drill a hole through the model. False positive pixels (i.e. those labelled as part of the object while they should be labelled as background) will keep as opaque a voxel where there is nothing. Most probably, these false opaque voxels will be removed by other views of the object. The probability that a given empty voxel be labelled as opaque for all the views is very low.

We present results of 3D modelling with shape from silhouette because it is extremely sensitive to the camera location. Any error in the estimated position or orientation of the camera would result in removing a significant part of the voxels, that should not have been removed if the position of the camera had been computed accurately. For this reason, shape from silhouette is often used in the context of a single camera, where the object is rotated by a turntable, such that its movement can be calibrated by an independent system. In our case, the stereo setup will provide the motion of the cameras. Furthermore, the use of color images is very useful, since it makes background extraction very easy to perform. In our case, we used black and white images, which means the task of background extraction (a basic operation in shape from silhouette implementation) had to be done by correlation of windows.

Figure 6.2: Samples of the *Running Shoe* sequence, as captured by the right camera

6.1.1 Experimental Results

Figure 6.2 shows samples of the *Running Shoe* sequence. The projection matrices were corrected using the linear interpolation of the correction matrix parameters described in section 4.7.1. The silhouettes were built through correlation of windows surrounding a pixel of interest in the scene image with the background image.

Figure 6.3 shows an example of the background extraction results. This figure emphasizes the difficulties encountered in this task, when using large correlation windows (to compensate the lack of color information): the rounding effect is due to the use of windows (in this case 9×9 pixels); False negative "holes" are present in the middle of the silhouette because the object gray level is too close to the background's; False positive "snow" appears everywhere around the object.

(a) (b) (c)

Figure 6.3: (a) Background for the right camera of the *Running Shoe* sequence (b) Sample of the *Running Shoe* sequence (c) Extracted silhouette with three distinctive features: rounding effect by the use of correlation windows, false negative "holes" and false positive "snow"

Figure 6.4 shows the 3D model obtained through silhouette intersection. Each voxel is a cube whose side is 5 mm.

Figure 6.5 shows samples of the *Duck* sequence. The projection matrices were corrected using the linear interpolation of the correction matrix parameters described in section 4.7.1. A few views of the obtained 3D model are shown in Figure 6.6. Each cubic voxel has a side of 5 mm.

Figure 6.7 shows another model built from the *Duck* sequence, with voxels of 2 mm. It emphasizes a number of holes transpiercing the volume, some being real (the object has two holes transpiercing the body, cf. Figure 6.5), some being the result of bad background extraction. This is an example of the catastrophic impact of a group of pixels falsely identified as part of the background.

6.2 Augmented Reality

An augmented reality application is a system in which computer-generated data is supplied to the user, in addition to its own perception of the real world [20], [21]. This data can be audio, video (text, map, moving picture) or kinesthetic. The important point is that it must be superimposed with real-world data, either directly acquired by the user or through sensors. In the case one wants to augment reality with 3D virtual

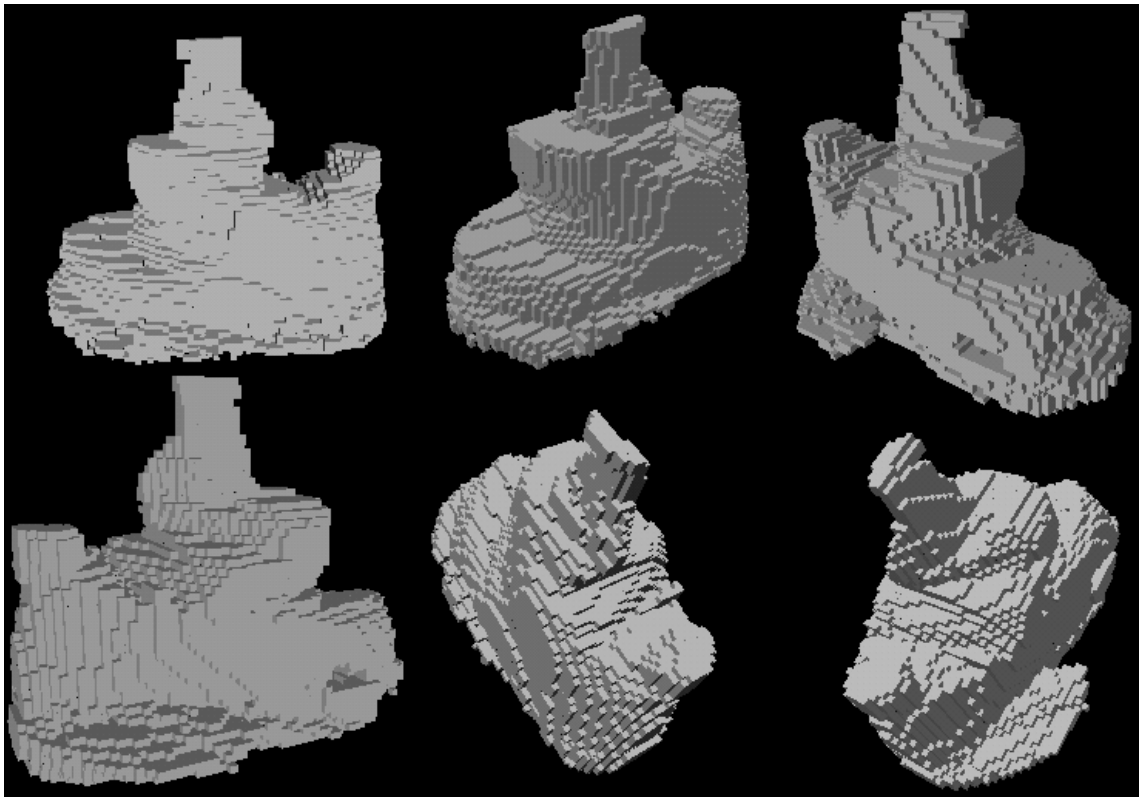


Figure 6.4: Views of the *Running Shoe* model

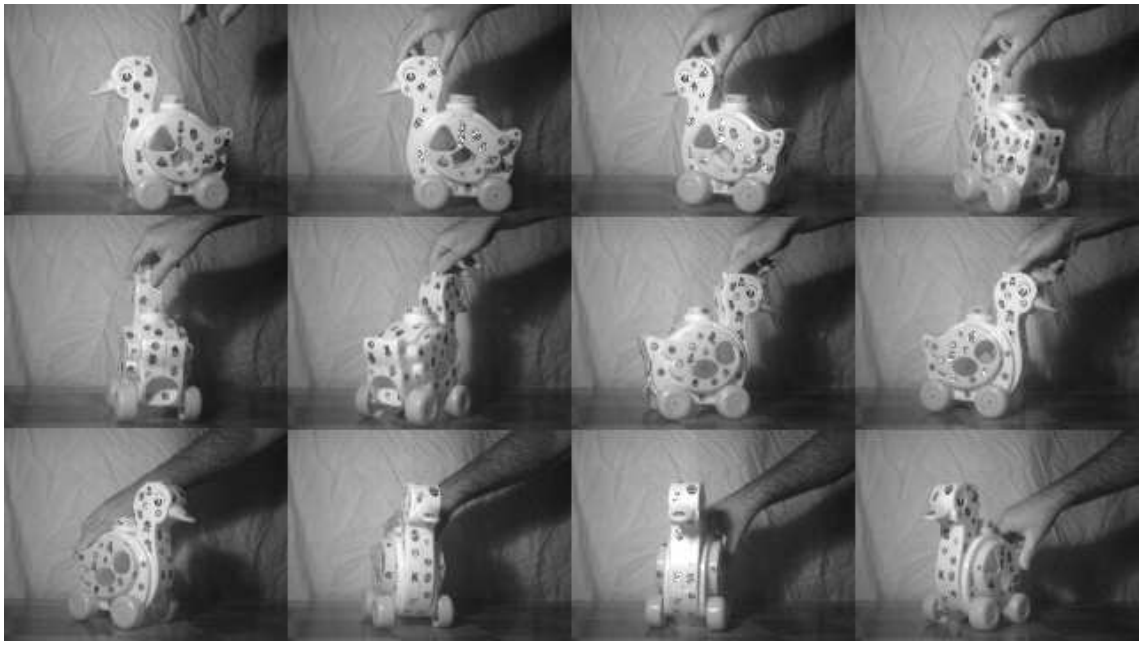


Figure 6.5: Samples of the *Duck* sequence, as captured by the left camera

objects, the registration between the real-world reference frame and the virtual object reference frame must be maintained. The camera pose estimation from a stereo setup can be used to keep track of the position of the cameras, and therefore project the virtual object in the images in a realistic way.

6.2.1 Experimental Results

Figure 6.8 shows samples of the *Duck* sequence, over which the axes of a reference system attached to the object have been drawn, according to the raw projection matrices. The reference system was arbitrarily positioned with its origin inside the object volume, with its axes parallel to the object surfaces ($Z-$axis pointing up). The axes of the reference system can be displayed on the image by drawing segments whose endpoints are the reference system origin and unity vectors in the principal directions ($[0,0,L]^T, [0,L,0]^T, [L,0,0]^T$). Once it is proved that a reference frame can be inserted in an image, any other virtual object can easily be added. It means that the registration between the cameras and the object is preserved along the sequence,

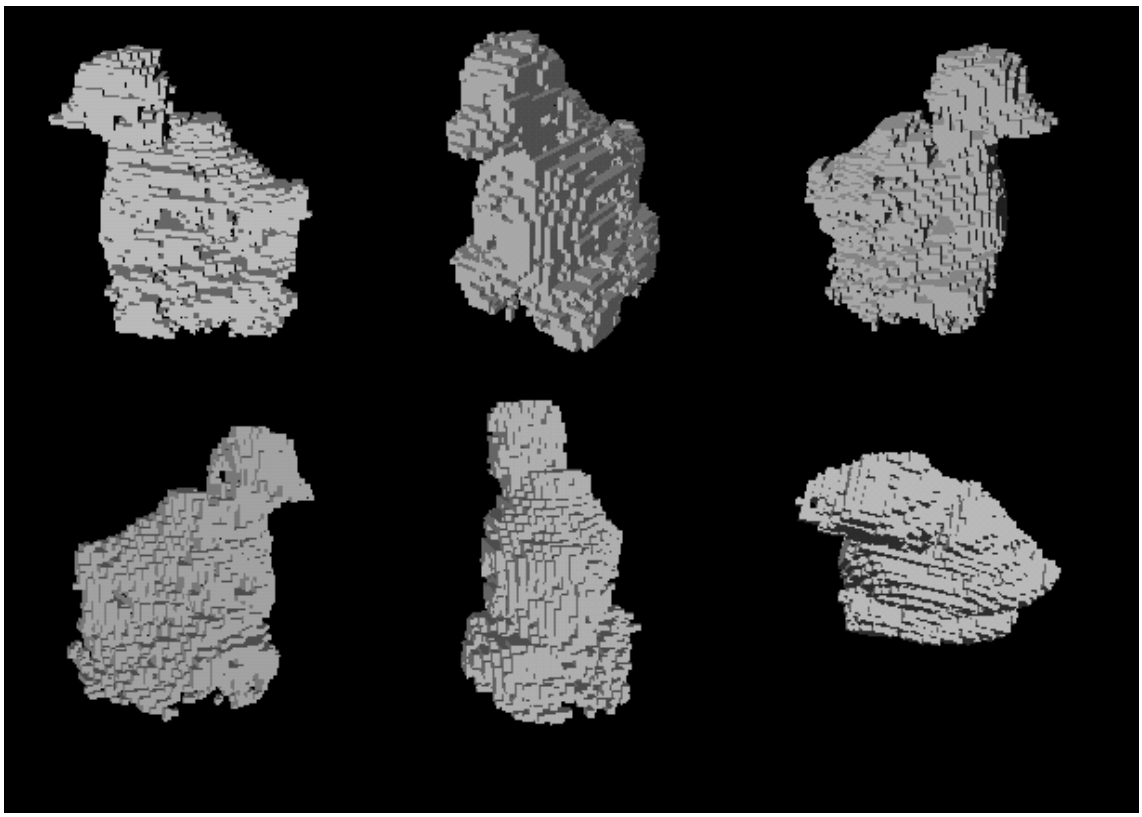


Figure 6.6: Views of the *Duck* model

Figure 6.7: A view of the *Duck* model, with a resolution of 2 mm

allowing realistic display of virtual objects in the real scene.

From Figure 6.8, it can clearly be observed that one of the registrations between the 9^{th} displayed capture and the 10^{th} displayed capture is not accurate (one out of three images are displayed, hence there are two more captures in between these two). Of course, since every new position is computed with respect to the previous one, all the views after the 9^{th} displayed capture show an obvious misalignment of their reference frame. Although it is easy to pinpoint which registration step is wrong through visual inspection, it is extremely difficult to achieve through an algorithmic procedure. We decided not to investigate such a research path, and limited ourself to the uniform error distribution along the sequence.

Figure 6.9 shows the same sequence with the axes of its attached reference frame, after a loop has been detected and after having linearly interpolated the correction matrix parameters. The drawn axes remain reasonably fixed with respect to the object, indicating an overall gain in the tracking of the object location. Although we

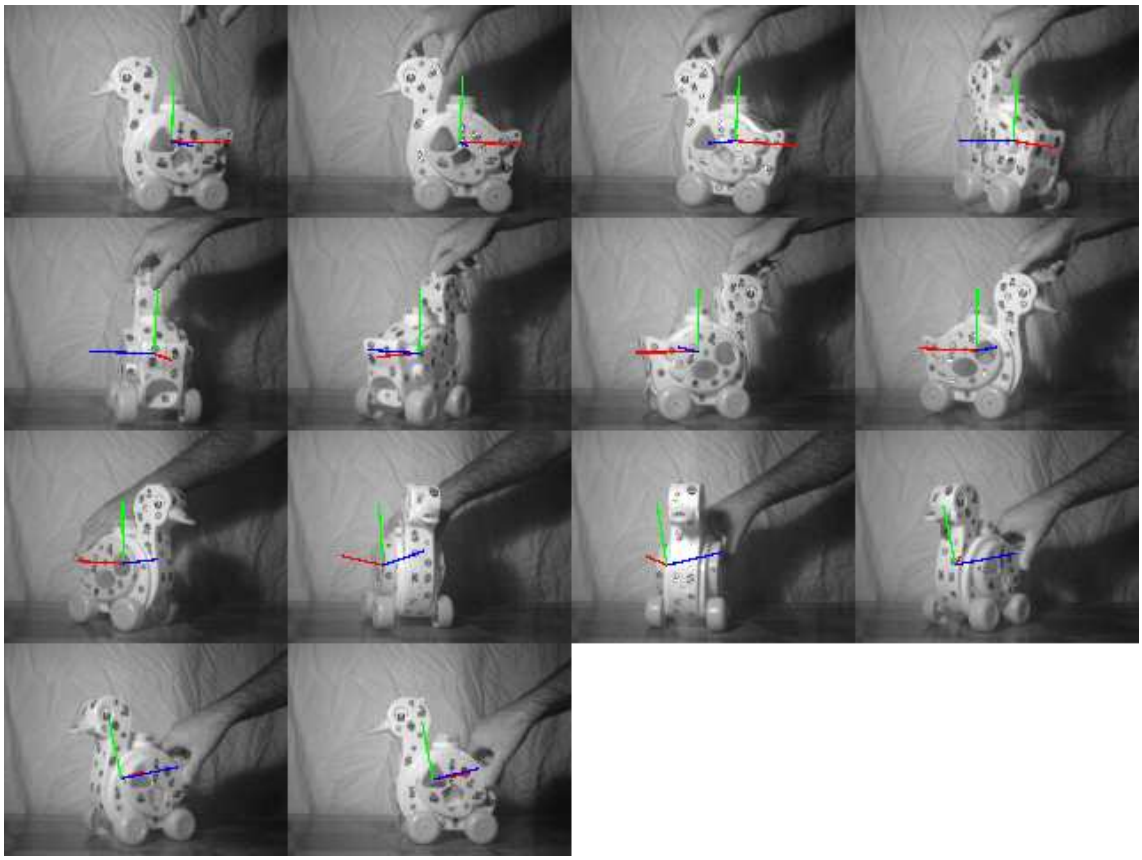


Figure 6.8: Samples of the *Duck* sequence with an attached reference system, before correction of the projection matrices

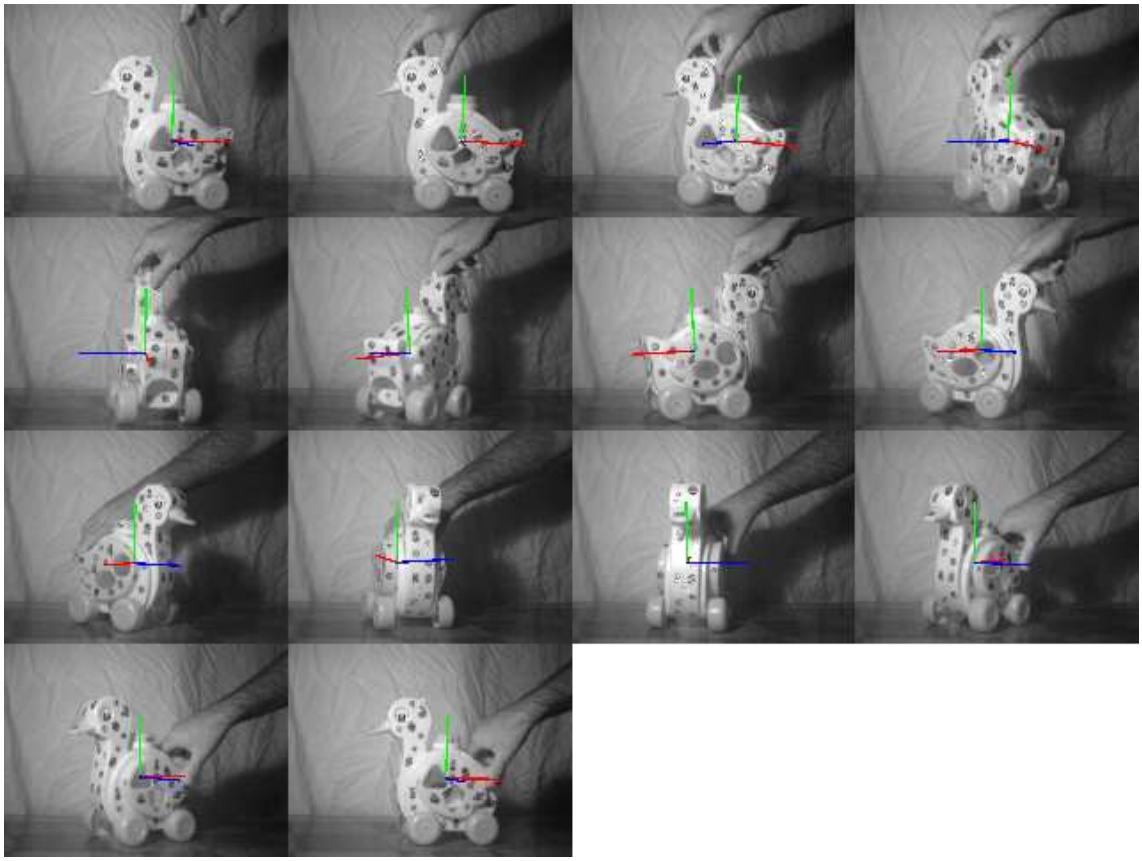


Figure 6.9: Samples of the *Duck* sequence with an attached reference system, after correction of the projection matrices

know that most of the error was due to some specific registration steps, we distributed the registration error along the whole sequence, and no obvious break occurs in the perceived accuracy of the axes location.

Figure 6.10 shows the *Russian Headstock* sequence with the axes of a reference system fixed to the base of the object. The projection matrices were corrected through a linear interpolation of the correction matrix parameters. The equivalent sequence, before error correction, does not show a significant difference since the error accumulation was very low for this sequence.

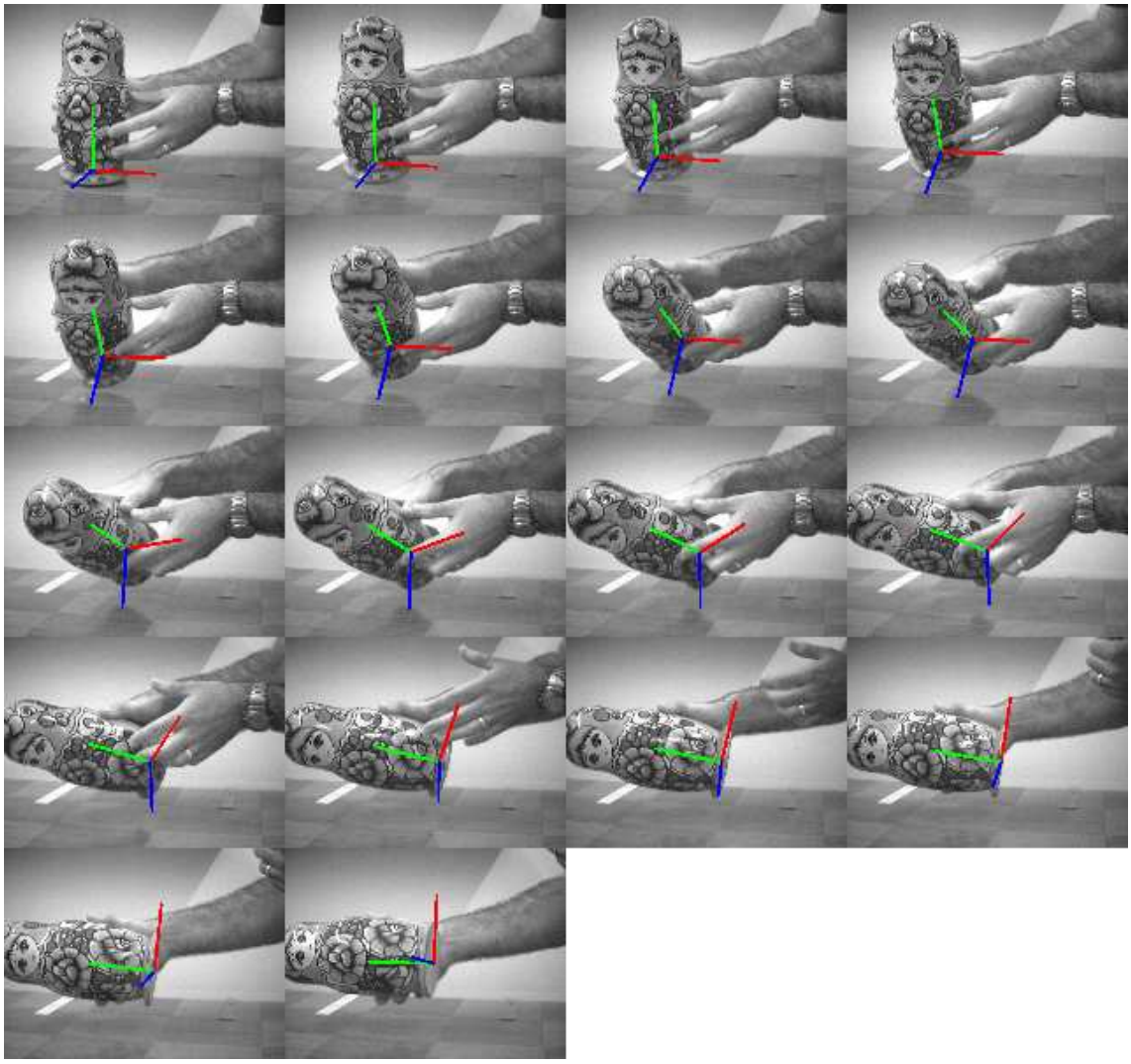


Figure 6.10: *Russian Headstock* sequence with an attached reference system, after correction of the projection matrices

Chapter 7

Conclusion

In this thesis, we addressed the problem of camera pose estimation, which is equivalent to the problem of 3D registration of a rigid object moving in front of the cameras. We used a calibrated stereoscopic vision setup to track the camera positions with respect to a moving rigid object, along image sequences. We reviewed the mathematics of rigid transformations in space and the geometry of the stereoscopic setup. We showed that the cameras must be calibrated independently and the fundamental matrix constructed from the calibration parameters, without invoking the eight-point algorithm. We proposed a robust 3D registration procedure that exploits the rigidity of the scene to automatically filter out the reconstructed points originating from false matches and errors in feature tracking. An error correction scheme was introduced, which takes advantage of loops in the movement of the cameras to compensate for the accumulated error. Comparison with a bundle adjustment commercial software showed that the correction scheme brings the error to a baseline level of disagreement between the two techniques. Through experimental results, we showed the validity of the obtained projection matrices and that their accuracy was sufficient for tasks such as model building or scene augmentation.

7.1 Difficulties and Problems

For technical reasons, the experiments were conducted with black and white images, which caused a lot of difficulties, especially for the task of background extraction. It would force us to use correlation windows instead of simple single pixel color comparison. It is believed that the introduction of color images would improve both experimental results and speed.

The goal of the present thesis was to provide a feasibility study of the proposed method. Some applications, especially those involving tool tracking, require real-time processing, which was not the case in the actual implementations. Table 7.1 shows some typical time values of the actual implementations, setting the parameters such that the initial number of feature matches would be around 1000.

It can be observed that most of the time was spent on matching and tracking tasks. Although libraries exist which provide implementations of these functions, we chose to implement them ourself, in order to have a limited number of well understood parameters. This fact can explain the obvious inefficiency of these steps, which can be performed in real-time by optimized code, perhaps with a smaller number of feature points. Thus, the first possible improvement toward faster execution would be to replace the matching and tracking functions by optimized code, after having tuned all the parameters by trials and errors.

The reconstruction and the robust registration steps are very sensitive to the number of tracked feature points. Thus, the second possible improvement would be to set the matching function parameters such that the number of feature points returned is around 20 (instead of 1000).

7.2 Future Work

In addition to the most obvious improvements to the implementation code (use of color images, real-time video), some applications of the proposed method can be investigated further. Among the most promising ones, let us mention tool tracking

Table 7.1: Typical time values of the functions actually implemented

Step	Typical time per execution	Typical number of executions per registration	Typical time per registration
Matching	90 s	1	90 s
Tracking	340 s	2	680 s
3D Reconstruction	7×10^{-5} s	2000	0.14 s
Robust registration	6 s	1	6 s
Computation of the new camera positions	1×10^{-6} s	2	2×10^{-6} s
Detection of previously viewed locations	9×10^{-5} s	40	0.0036 s
Matching, tracking and robust registration between non-consecutive captures	130 s	2	260 s
Accumulated error correction	4×10^{-6} s	2	8×10^{-6} s

and robotics.

7.2.1 Tool Tracking

In many simulations and training applications, a user has to freely manipulate a tool. For instance, in the medical area, virtual reality training requires the apprentice to handle surgical instruments while performing a simulated operation. Furthermore, some commercial software exist to guide the surgeon during a real-world surgery by tracking the position of their instrument [1]. In military training, an immersive virtual reality system can be built to simulate a combat situation, in which the precise location of the soldier's weapon must be tracked.

The previous section showed experimentally that the proposed technique can be used to track the 3D location of an object with respect to a global reference frame. The main additional constraint is the real-time requirement of a tracking system. In this scheme, when a loop is detected, the accumulated error can be compensated for, but there is no need to correct the intermediate matrices. Therefore, the error in the position of the tool would accumulate with time, and would be reset each time the tool gets back into a position such that the Z-axis of the cameras would be parallel and close enough to the Z-axis of the cameras at the time of first capture (c.f. section 4.5).

7.2.2 Robotics

A common problem in robotics applications is the error accumulation in the recorded position of a freely moving robot or a robotic arm. Due to a large number of factors (inaccuracy in parts manufacturing, delays, temperature changes, etc.), the encoders embedded in a robotic system accumulate error after having been calibrated. For some critical applications, calibration must be done regularly. Camera pose estimation from a stereo setup mounted on the robot may help simplify the recalibration procedure.

[1] www.orthosoft.ca

The cameras must record images at the starting position, after the stereo setup has been calibrated. Then, when the robot needs to recalibrate, it orients its stereo setup in the direction of the initial capture. As depicted in section 4.7, the error in the position can then be corrected, resetting the accumulated error. This scenario would be applicable in situations where the resultant position of the stereo head is more important than the individual state of all the joints of the robot.

Bibliography

[1] Trucco E., Verri A. 1998. Introductory Techniques for 3-D Computer Vision, Prentice-Hall (eds)

[2] Longuet-Higgins H.C. "A Computer Algorithm for Reconstructing a Scene from Two Projections", in *Nature*, vol. 293, no. 10, September 1981

[3] Hartley R.I. "In Defense of the Eight-Point Algorithm", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, June 1997

[4] Tsai Roger Y. "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", in *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, August 1987

[5] Zhang Zhengyou "A Flexible New Technique for Camera Calibration", Microsoft Research Technical Report MSR-TR-98-71, December 1998

[6] Malm Henrik, Heyden Anders "Stereo Head Calibration from a Planar Object", in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, December 2001, pp II-657 - II-662

[7] Arun K.S., Huang T.S., Blostein S.D. "Least-Squares Fitting of Two 3-D Point Sets", in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, Sept. 1987

[8] Fischler Martin A., Bolles Robert C. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", in *Communications of the ACM*, vol. 24, no. 6, June 1981, pp 381-395

[9] Ramanathan Prashant, Steinbach Eckehard, Girod Bernd "Silhouette-based Multiple-View Camera Calibration", in *Proc. Vision, Modeling and Visualization 2000*, November 2000, pp 3-10

[10] Seo Y., Hong K.S. "Theory and Practice on the Self-Calibration of a Rotating and Zooming Camera from Two Views", in *IEE Proc. Vision and Image Signal Processing*, vol. 148, no. 3, June 2001, pp 166-172

[11] Laurentini Aldo "The Visual Hull Concept for Silhouette-Based Image Understanding", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, February 1994, pp 150-162

[12] Kuzu Yasemin, Rodehorst Volker "Volumetric Modeling Using Shape From Silhouette", in *Proc. of the Fourth Turkish-German Joint Geodetic Days*, April 2001, pp 469-476

[13] Jones M., Oakley J.P. "Efficient Representation of Object Shape for Silhouette Intersection", in *IEE Proc.-Vis. Image Signal Process.*, vol. 142, no. 6, December 1995, pp 359-365

[14] Matusik Wojciech, Buehler Chris, Raskar Ramesh, Gortler Steven J., McMillan Leonard "Image-Based Visual Hulls", in *Proceedings of SIGGRAPH 2000*, pp 369-374

[15] Snow Dan, Viola Paul, Zabih Ramin "Exact Voxel Occupancy with Graph Cuts", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, vol. 3, 2000, pp 345-352

[16] Memon Qurban, Khan Sohaib "Camera Calibration and Three-Dimensional World Reconstruction of Stereo-Vision Using Neural Networks", in *International Journal of Systems Science*, vol. 32, no. 9, 2001, pp 1155-1159

[17] Do Yongtae "Application of Neural Network for Stereo-Camera Calibration", in *Proc. of the 1999 IEEE/RJS International Conference on Intelligent Robots and Systems*, pp 2719-2722

[18] Faugeras Olivier, Luong Quang-Tuan 2001. The Geometry of Multiple Images, The MIT Press (eds)

[19] Bartoli Adrien "A Unified Framework for Quasi-Linear Bundle Adjustment", in *Proc. of the 16th International Conference on Pattern Recognition, vol.2, 2002*, pp 560-563

[20] You Suya, Neumann Ulrich, Azuma Ronald "Orientation Tracking for Outdoor Augmented Reality Registration", in *IEEE Computer Graphics and Applications, vol. 19, no. 6, 1999*, pp 36-42

[21] Neumann Ulrich, Cho Youngkwan "A Self-Tracking Augmented Reality System", in *ACM International Symposium on Virtual Reality and Applications, July 1996*, pp 109-115

[22] Roth Gerhard "Computing Camera Positions from a Multi-Camera Head", in *Proc. IEEE 3rd International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 135-142

[23] Zhang Zhengyou "Motion and Structure of Four Points from One Motion of a Stereo Rig with Unknown Extrinsic Parameters", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, December 1995, pp. 1222-1227

[24] Ho Pui-Kuen, Chung Ronald "Stereo-Motion with Stereo and Motion in Complement", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, February 2000, pp. 215-220

[25] Vincent Étienne "On Feature Point Matching, in the Calibrated and Uncalibrated Contexts, between Widely and Narrowly Separated Images", Ph.D. thesis, Ottawa-Carleton Institute for Computer Science, School of Information Technology and Engineering, University of Ottawa, Ottawa, 2004

[26] Pritchett Philip, Zisserman Andrew "Wide Baseline Stereo Matching", in *Proc. 6th International Conference on Computer Vision (ICCV'1998)*, 1998, pp. 754-760

[27] Laganière Robert, Vincent Étienne "Wedge-Based Corner Model for Widely Separated Views Matching", in *Proc. International Conference on Pattern Recognition (ICPR'2002)*, vol II, 2002, pp. 856-860

[28] Tuytelaars Tinne, Van Gool Luc "Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions", in *Proc. 11th British Machine Vision Conference (BMVC'2000)*, 2000

[29] Tell Dennis, Carlsson Stefan "Combining Appearance and Topology for Wide Baseline Matching", in *Proc. 7th European Conference on Computer Vision - Part I*, 2002, pp. 68-81

[30] Baumberg A. "Reliable Feature Matching Across Widely Separated Views", in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, vol. 1, 2000, pp. 774-781

[31] Vincent Étienne, Laganière Robert "Matching Feature Points in Stereo Pairs: A Comparative Study of Some Matching Strategies", in *Machine Graphics and Vision (MGV'2001)*, vol. 10, no. 3, 2001, pp. 237-259

[32] Kanade Takeo, Okutomi Masatoshi "A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment", in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, September 1994, pp 920-932

[33] Tomasi Carlo, Manduchi Roberto "Stereo Matching as a Nearest-Neighbor Problem", in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, March 1998, pp 333-340

[34] Candocia Frank, Adjouadi Malek "A Similarity Measure for Stereo Feature Matching", in *IEEE Trans. on Image Processing*, vol. 6, no. 10, October 1997, pp 1460-1464

[35] Han Kyu-Phil, Song Kun-Woen, Chung Eui-Yoon, Cho Seok-Je, Ha Yeong-Ho "Stereo Matching Using Genetic Algorithm with Adaptive Chromosomes", in *Pattern Recognition 34*, 2001, pp. 1729-1740

[36] Han Youngmo "Relations Between Bundle-Adjustment and Epipolar-Geometry-Based Approaches, and their Applications to Efficient Structure from Motion", in *Real-Time Imaging 10*, 2004, pp. 389-402

[37] Dornaika F., Chung R. "Stereo Correspondence from Motion Correspondence", in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 1999, pp. 70-75

[38] Hanna K., Okamoto N. "Combining Stereo and Motion Analysis for Direct Estimation of Scene Structure", in *Proc. 4th International Conf. on Computer Vision*, 1993, pp. 357-365

[39] Dornaika F., Chung R. "Cooperative Stereo-Motion: Matching and Reconstruction", in *Computer Vision and Image Understanding*, no. 79, no. 3, 2000, pp. 408-427

[40] Strecha Christoph, Van Gool Luc "Motion - Stereo Integration for Depth Estimation", in *Proc. 7th European Conf. on Computer Vision*, part II, 2002, pp. 170-185

[41] Stein Gideon, Shashua Amnon "Direct Estimation of Motion and Extended Scene Structure from a Moving Stereo Rig", in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1998

[42] Ji Qiang, Zhang Yongmian "Camera Calibration with Genetic Algorithms", in *IEEE Trans. on Systems, Man and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 2, 2001, pp. 120-130

[43] Roberts M., Naftel A.J. "A genetic algorithm approach to camera calibration in 3D machine vision", in *IEE Colloquium on Genetic Algorithms in Image Processing and Vision*, 1994, pp. 12/1 - 12/5

[44] Ahmed M., Hemayed E., Farag A. "Neurocalibration: A Neural Network That Can Tell Camera Calibration Parameters", in *Proc. IEEE International Conference on Computer Vision - Volume 1*, 1999

[45] Zisserman Andrew, Beardsley Paul A., Reid Ian D. "Metric Calibration of a Stereo Rig", in *Proc. IEEE Workshop on Representation of Visual Scenes*, 1995, pp. 93-100

[46] Zhang Zhengyou, Faugeras Olivier "An Effective Technique for Calibrating a Binocular Stereo Through Projective Reconstruction Using Both a Calibration Object and the Environment", in *Videre: Journal of Computer Vision Research*, vol. 1, no. 1, 1997, pp. 58-68

[47] Brooks M.J., de Agapito L., Huynh D.Q., Baumela L. "Direct Methods for Self-Calibration of a Moving Stereo Head", in *Proc. 4th European Conf. on Computer Vision*, vol. II, 1996, pp. 415-426

[48] Ruf Andreas, Csurka Gabriella, Horaud Radu "Projective Translations and Affine Stereo Calibration", in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, pp. 475-481

[49] Horn Berthold Klaus Paul. 1986. Robot Vision, MIT Press (eds)

[50] Faugeras Olivier. 1993. Three-Dimensional Computer Vision, a Geometric Viewpoint, MIT Press (eds)

Appendix A

Transformations in Space

Coordinates in space are data relative to a reference system. A single point x may have coordinates $\vec{x}_{table}$ with respect to a table, $\vec{x}_{corner}$ with respect to the corner of the room, and $\vec{x}_{world}$ with respect to a third global reference frame. The way to preserve the consistency of these valid representations is through the use of *homogeneous transformation matrices*. They are 4×4 matrices built with two basic objects: a rotation matrix and a translation vector. Alternatively, they can be used to represent the movement of a solid object. In this scheme, the transformation is applied to compute the new location of each of the points of the object, in the same reference system, after rotation and translation took place. This chapter introduces the conventions that will be used in this thesis, along with important results and properties of homogeneous transformation matrices.

A.1 Rotation Matrix

To completely describe a three-dimensional rotation, one needs to supply the values of three angles. For instance, these may be the azimuth, the elevation and the spin. This representation is visually simple, but mathematically complicated. A better way of expressing a rotation in space would be the following: first, rotate the set of axes around the X-axis by an angle ψ. Then, rotate the obtained set of axes around

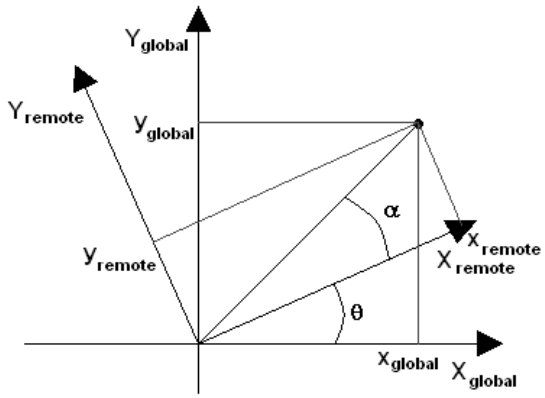


Figure A.1: Rotation of two reference systems around the Z axis

its own Y-axis by an angle ϕ. Finally, rotate the obtained set of axes around its own Z-axis by an angle θ. The triplet (θ, ϕ, ψ) are called the *Euler angles*. They don't relate easily to the more visually attractive azimuth, elevation and spin, but the corresponding rotation matrix is easy to construct.

This way of defining the rotation from the three elementary angles is arbitrary. One could have chosen to first apply a rotation around the Z-axis, followed by a rotation along the obtained Y-axis, followed by a rotation around the obtained X-axis, and still get a valid rotation matrix, although with a different analytical expression.

Let us state the expression of the rotation matrix, as a function of the Euler angles.

$$R(\theta,\phi,\psi) = \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \end{bmatrix} \tag{A.1}$$

Proof. Let us assume we have two sets of orthonormal axes called *global* and *remote*. They share the same origin, but they are rotated one with respect to the other

by an angle θ in the $X-Y$ plane (see Figure A.1). A point having coordinates (x_{remote}, y_{remote}) in the remote reference system will have (ρ, θ) polar coordinates:

$$\begin{bmatrix} x_{remote} \\ y_{remote} \\ z_{remote} \end{bmatrix} = \begin{bmatrix} \rho\cos\alpha \\ \rho\sin\alpha \\ z_{remote} \end{bmatrix} \tag{A.2}$$

The same point in space, expressed in the global reference system, can be written:

$$\begin{bmatrix} x_{global} \\ y_{global} \\ z_{global} \end{bmatrix} = \begin{bmatrix} \rho\cos(\alpha+\theta) \\ \rho\sin(\alpha+\theta) \\ z_{remote} \end{bmatrix} = \begin{bmatrix} \rho\cos\alpha\cos\theta - \rho\sin\alpha\sin\theta \\ \rho\cos\alpha\sin\theta + \rho\sin\alpha\cos\theta \\ z_{remote} \end{bmatrix} \tag{A.3}$$

Inserting (A.2) in (A.3):

$$\begin{bmatrix} x_{global} \\ y_{global} \\ z_{global} \end{bmatrix} = \begin{bmatrix} x_{remote}\cos\theta - y_{remote}\sin\theta \\ x_{remote}\sin\theta + y_{remote}\cos\theta \\ z_{remote} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{remote} \\ y_{remote} \\ z_{remote} \end{bmatrix} \tag{A.4}$$

Let us define $R(\theta, 0, 0)$ the rotation matrix around the Z axis by an angle θ (the RPY convention is used, i.e. the *roll* is quoted first, followed by the *pitch* and the *yaw*):

$$R(\theta, 0, 0) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

Then, by symmetry:

$$R(0, \phi, 0) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{A.6}$$

$$R(0, 0, \psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{A.7}$$

where ϕ and ψ are the rotation angles around the Y and the X axis, respectively. The rotation matrix can be computed by the cascade of a rotation of ψ around the X axis, premultiplied by a rotation of ϕ around the Y axis, premultiplied by a rotation of θ around the Z axis, that is:

$$R(\theta, \phi, \psi) = R(\theta, 0, 0)R(0, \phi, 0)R(0, 0, \psi) \tag{A.8}$$

That can be rewritten as:

$$R(\theta, \phi, \psi) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{A.9}$$

and resulting in:

$$R(\theta, \phi, \psi) = \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \end{bmatrix} \tag{A.10}$$

□

A.1.1 Dual Angular Representation

When using the convention of angular representation described above, two sets of angles will yield to the same rotation matrix. This must be taken into account whenever one wants to compute the constituting angles from a given rotation matrix. It is indeed easy to show that:

$$R(\theta - \pi, \pi - \phi, \pi + \psi) = R(\theta, \phi, \psi) \tag{A.11}$$

A.1.2 Orthogonality of a Rotation Matrix

The inverse of a rotation matrix is its transpose. As a consequence, the rotation matrix is orthogonal.

$$R^{-1} = R^T \tag{A.12}$$

Proof.

$$R^T(\theta,\phi,\psi)R(\theta,\phi,\psi) =$$
$$\begin{bmatrix} \cos\theta\cos\phi & \sin\theta\cos\phi & -\sin\phi \\ \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \cos\phi\sin\psi \\ \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi & \cos\phi\cos\psi \end{bmatrix}$$
$$\begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.13}$$

Therefore:

$$R^T = R^{-1} \tag{A.14}$$

□

The transpose of a rotation matrix is another rotation matrix.

Proof. Since $R^T R = I$, the effect of premultiplying the transpose of a rotation matrix to a rotated vector cancels out the effect of the rotation. In other words, it is another rotation in the opposite direction. □

This result is especially useful when demonstrating properties of rows or columns of a rotation matrix. If one demonstrates a given property of the rows of a rotation matrix, the same property will hold for the columns, and vice-versa.

A.1.3 Orthogonality of the Columns and the Rows Vectors

Let $\vec{r}_k$ be the vector built with the entries of the k^{th} row of a rotation matrix $R(\theta, \phi, \psi)$. Let $\vec{c}_k$ be the vector built with the entries of the k^{th} column of a rotation matrix $R(\theta, \phi, \psi)$. Then,

$$\vec{r}_i \cdot \vec{r}_j = \delta[i-j] \tag{A.15}$$

$$\vec{c}_i \cdot \vec{c}_j = \delta[i-j] \tag{A.16}$$

Proof. From (A.12), we can write:

$$RR^T = I \tag{A.17}$$

$$\begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \\ \vec{r}_3 \end{bmatrix} \begin{bmatrix} \vec{r}_1^T & \vec{r}_2^T & \vec{r}_3^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.18}$$

$$\vec{r}_i \cdot \vec{r}_j = \delta[i-j] \tag{A.19}$$

and reciprocally:

$$R^T R = I \tag{A.20}$$

$$\begin{bmatrix} \vec{c}_1^T \\ \vec{c}_2^T \\ \vec{c}_3^T \end{bmatrix} \begin{bmatrix} \vec{c}_1 & \vec{c}_2 & \vec{c}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.21}$$

$$\vec{c}_i \cdot \vec{c}_j = \delta[i-j] \tag{A.22}$$

□

A.1.4 Cross Product of the Rows or the Columns Vectors

Let $\vec{r}_k$ be the column vector built with the entries of the k^{th} row (or column) of a rotation matrix. Then,

$$\vec{r}_1 = \vec{r}_2 \times \vec{r}_3 \tag{A.23}$$

$$\vec{r}_2 = \vec{r}_3 \times \vec{r}_1 \tag{A.24}$$

$$\vec{r}_3 = \vec{r}_1 \times \vec{r}_2 \tag{A.25}$$

Or, in a more compact formulation:

$$\vec{r}_\alpha = \vec{r}_\beta \times \vec{r}_\gamma \tag{A.26}$$

where (α, β, γ) is an even permutation of $(1, 2, 3)$.

Proof. Let us use the row vectors for the demonstration.

$$\vec{r}_2 \times \vec{r}_3 =$$

$$\det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi \\ \sin\theta\sin\psi + \cos\theta\sin\phi\cos\psi \end{bmatrix}$$

$$= \vec{r}_1 \tag{A.27}$$

$$\vec{r}_3 \times \vec{r}_1 =$$

$$\det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \\ \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \end{bmatrix}$$

$$= \begin{bmatrix} \sin\theta\cos\phi \\ \cos\theta\cos\psi + \sin\theta\sin\phi\sin\psi \\ \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \end{bmatrix}$$

$$= \vec{r}_2 \tag{A.28}$$

$$\vec{r}_1 \times \vec{r}_2 =$$

$$\det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \end{bmatrix}$$

$$= \begin{bmatrix} -\sin\phi \\ \cos\phi\sin\psi \\ \cos\phi\cos\psi \end{bmatrix}$$

$$= \vec{r}_3 \tag{A.29}$$

□

A.1.5 Eigenvalues of a Rotation Matrix

The three eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of $R(\theta, \phi, \psi)$ are

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{d(\theta,\phi,\psi)-1+\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} \\ \frac{d(\theta,\phi,\psi)-1-\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} \end{bmatrix} \tag{A.30}$$

where

$$d(\theta,\phi,\psi) = \cos\theta\cos\phi + \cos\theta\cos\psi + \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi \tag{A.31}$$

Proof. We are searching for the values of λ that will satisfy

$$\det(\lambda I - R) = 0 \tag{A.32}$$

$$\det\begin{bmatrix} \lambda - \cos\theta\cos\phi & \sin\theta\cos\psi - \cos\theta\sin\phi\sin\psi & -\cos\theta\sin\phi\cos\psi - \sin\theta\sin\psi \\ -\sin\theta\cos\phi & \lambda - \sin\theta\sin\phi\sin\psi - \cos\theta\cos\psi & -\sin\theta\sin\phi\cos\psi + \cos\theta\sin\psi \\ \sin\phi & -\cos\phi\sin\psi & \lambda - \cos\phi\cos\psi \end{bmatrix} = 0 \tag{A.33}$$

$$\begin{gathered}(\lambda - \cos\theta\cos\phi)[\lambda^2 - \lambda\cos\phi\cos\psi - \lambda\sin\theta\sin\phi\sin\psi + \sin\theta\sin\phi\cos\phi\sin\psi\cos\psi \\ -\lambda\cos\theta\cos\psi + \cos\theta\cos\phi\cos^2\psi - (\sin\theta\sin\phi\cos\phi\sin\psi\cos\psi - \cos\theta\cos\phi\sin^2\psi)] \\ -(\sin\theta\cos\psi - \cos\theta\sin\phi\sin\psi)[-\lambda\sin\theta\cos\phi + \sin\theta\cos^2\phi\cos\psi \\ -(-\sin\theta\sin^2\phi\cos\psi + \cos\theta\sin\phi\sin\psi)] \\ +(-\cos\theta\sin\phi\cos\psi - \sin\theta\sin\psi)[\sin\theta\cos^2\phi\sin\psi \\ -(\lambda\sin\phi - \sin\theta\sin^2\phi\sin\psi - \cos\theta\sin\phi\cos\psi)] \\ = 0\end{gathered} \tag{A.34}$$

$$\begin{gathered}\lambda^3 - \lambda^2[\cos\theta\cos\phi + \cos\theta\cos\psi + \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi] \\ +\lambda[\cos\theta\cos\phi + \cos\theta\cos\psi + \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi] - 1 \\ = 0\end{gathered} \tag{A.35}$$

$$\lambda^3 - d(\theta,\phi,\psi)\lambda^2 + d(\theta,\phi,\psi)\lambda - 1 = 0 \tag{A.36}$$

where

$$d(\theta,\phi,\psi) \equiv \cos\theta\cos\phi + \cos\theta\cos\psi + \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi \tag{A.37}$$

The characteristic equation (A.36) has the root $\lambda = 1$.

$$\begin{aligned}\lambda^3 - d(\theta,\phi,\psi)\lambda^2 + d(\theta,\phi,\psi)\lambda - 1 &= (\lambda-1)[\lambda^2 + \lambda(1-d(\theta,\phi,\psi)) + 1] \\ &= (\lambda-1)(\lambda - \tfrac{d(\theta,\phi,\psi)-1+\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2})\times \\ &\quad (\lambda - \tfrac{d(\theta,\phi,\psi)-1-\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2}) = 0 \end{aligned} \tag{A.38}$$

Hence, the eigenvalues of the rotation matrix are:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{d(\theta,\phi,\psi)-1+\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} \\ \frac{d(\theta,\phi,\psi)-1-\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} \end{bmatrix} \tag{A.39}$$

where

$$d(\theta,\phi,\psi) = \cos\theta\cos\phi + \cos\theta\cos\psi + \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi \tag{A.40}$$

□

The unitary eigenvalue of the rotation matrix corresponds to the situation where a vector lying in the axis of rotation is premultiplied by the rotation matrix, leaving the vector unaffected. Thus, the axis of rotation is the eigenvector corresponding to $\lambda = 1$ and can be computed by solving the non-trivial solution of the following homogeneous equation:

$$(R - I)\vec{a} = \vec{0} \tag{A.41}$$

The two other eigenvalues can be written $e^{\pm i\alpha}$, where α is the global rotation around the axis.

A.1.6 Determinant of a Rotation Matrix

An orthogonal matrix must have a determinant of 1 or -1. This can be seen by the defining property of orthogonal matrices:

$$\begin{aligned} A^{-1} &= A^T \\ \det(A^{-1}) &= \det(A^T) \\ \frac{1}{\det(A)} &= \det(A) \\ \det(A) &= \pm 1 \end{aligned} \tag{A.42}$$

In the case of a rotation matrix, its determinant must be 1. An orthogonal transformation matrix that would have a determinant of -1 would represent a *reflection*.

$$\det(R(\theta,\phi,\psi)) = 1 \tag{A.43}$$

Proof. We will use the identity

$$|A| \overset{(B.22)}{=} \prod_{i=1}^{n} \lambda_i$$

with the eigenvalues given by (A.30):

$$\begin{aligned} \det(R(\theta,\phi,\psi)) = 1 \times \tfrac{d(\theta,\phi,\psi)-1+\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} \times \\ \tfrac{d(\theta,\phi,\psi)-1-\sqrt{d^2(\theta,\phi,\psi)-2d(\theta,\phi,\psi)-3}}{2} = 1 \end{aligned} \tag{A.44}$$

□

A.2 Homogeneous Transformation Matrix

General homogeneous transformations are constituted of a rotation and a translation. These two quantities must be defined by the way they relate a *remote* reference system with a *global* reference system. By definition, we will set:

$$\vec{x}_{global} \equiv R\vec{x}_{remote} + \vec{T} \tag{A.45}$$

In order to allow the transformation to be expressed in a single matrix, *homogeneous coordinates* must be introduced. A homogeneous coordinates vector is a 4×1 column vector whose three first entries are the x, y and z components of a 3D point, and whose 4^{th} entry is 1. Homogeneous coordinates are written in uppercase. If $\vec{X}$ is the homogeneous coordinates representation of $\vec{x}$ in (A.45), then

$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{global} &= \begin{bmatrix} r_{00} & r_{01} & r_{02} & T_x \\ r_{10} & r_{11} & r_{12} & T_x \\ r_{20} & r_{21} & r_{22} & T_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{remote} \\ \vec{X}|_{global} &= Q_{remote/global}\vec{X}|_{remote} \end{aligned} \tag{A.46}$$

where

$$Q_{remote/global} \equiv \begin{bmatrix} r_{00} & r_{01} & r_{02} & T_x \\ r_{10} & r_{11} & r_{12} & T_x \\ r_{20} & r_{21} & r_{22} & T_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.47}$$

Homogeneous transformation matrices are especially useful when used in conjunction with *transformation graphs*. A transformation graph is a pictorial representation of the objects of the world at hand, along with their reference systems. Homogeneous transformations are depicted by arrows. The origin of the arrow is the global reference system of the transformation, while the head of the arrow is the remote reference system of the transformation.

In Figure A.2, a 3D point **x** will have homogeneous coordinates $\vec{X}|_{global}, \vec{X}|_{table}$ or $\vec{X}|_{camera}$, depending on the reference system used. The homogeneous transformation matrices $Q_{table/world}$ (the transformation matrix of the table with respect to the world), $Q_{cam/world}$ and $Q_{cam/table}$ define the relative positions of these objects. Notice the redundancy of the information supplied: $Q_{cam/table}$ can be computed from the two other matrices.

$$Q_{cam/table} = Q^{-1}_{table/world}Q_{cam/world}$$

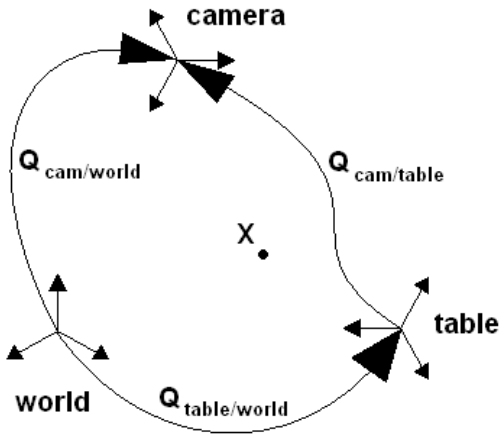


Figure A.2: A transformation graph

Proof.

$$\vec{X}|_{world} = Q_{table/world}\vec{X}|_{table} \Rightarrow \vec{X}|_{table} = Q^{-1}_{table/world}\vec{X}|_{world} \tag{A.48}$$

$$\vec{X}|_{world} = Q_{cam/world}\vec{X}|_{camera} \tag{A.49}$$

Inserting (A.49) into (A.48):

$$\vec{X}|_{table} = Q^{-1}_{table/world}Q_{cam/world}\vec{X}|_{camera} \tag{A.50}$$

$$Q_{cam/table} = Q^{-1}_{table/world}Q_{cam/world} \tag{A.51}$$

□

This simple example shows how one can follow the direction of the arrows (inverting the matrix when going in the opposite direction) to build transformation matrices relationships from inspection of a transformation graph.

Inverting a homogeneous transformation matrix is always possible. From (A.47), it can be seen that the determinant of a homogeneous transformation matrix equals the determinant of the rotation matrix, which is 1 (A.43).

A.3 Registration of Two Clouds of 3D Points

A key technique that will be used in this thesis is the ability to register two clouds of 3D points, that is, given two sets of corresponding points, to find the homogeneous transformation that makes the best overlap of the points. This method is described in [7]. In a first approach, we will derive the main results assuming that all the matches are good. In Chapter 4, we will introduce random sampling consensus (RANSAC) to get rid of the possible outliers.

A.3.1 Decoupling of the Optimal Rotation and the Optimal Translation

We will show that it is possible to find the optimal rotation $\tilde{R}$ independently, and then compute the optimal translation $\vec{\tilde{T}}$. Let us suppose there are n 3D points on a rigid object, for which we know their 3D location in two different positions of the object (see Figure A.3). Let $\{\vec{p}_i\}$ be the set of n 3D points in position 1 and let $\{\vec{p'}_i\}$ be the corresponding n 3D points in position 2, such that

$$\vec{p'}_i = R\vec{p}_i + \vec{T} + \vec{\epsilon}_i(R, \vec{T}) \tag{A.52}$$

where $\vec{\epsilon}_i(R, \vec{T})$ is the error vector associated with a particular choice of rotation matrix and translation vector, including some noise in the measurements. Let $\tilde{R}$ and $\vec{\tilde{T}}$ represent the optimal transformation that will minimize $\sigma^2(R, \vec{T})$, the sum of the squared norm of the error vectors:

$$\sigma^2(R, \vec{T}) \equiv \sum_{i=0}^{n-1} \parallel \vec{p'}_i - R\vec{p}_i - \vec{T} \parallel^2 \tag{A.53}$$

Let $\{\vec{p''}_i\}$ be the set of points obtained by applying the optimal transformation to the set of points in position 1:

$$\vec{p''}_i \equiv \tilde{R}\vec{p}_i + \vec{\tilde{T}} \tag{A.54}$$

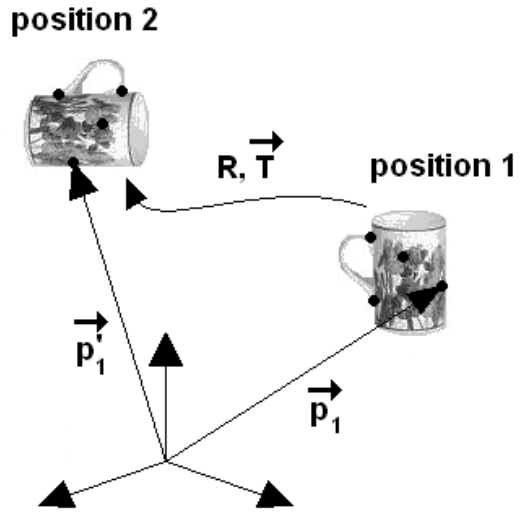


Figure A.3: Registration of two positions of a rigid object

Let $\vec{p}$, $\vec{p'}$ and $\vec{p''}$ be the centroids of the corresponding points sets:

$$\vec{p} \equiv \frac{1}{n}\sum_{i=0}^{n-1}\vec{p}_i \tag{A.55}$$

$$\vec{p'} \equiv \frac{1}{n}\sum_{i=0}^{n-1}\vec{p'}_i \tag{A.56}$$

$$\vec{p''} \equiv \frac{1}{n}\sum_{i=0}^{n-1}\vec{p''}_i \tag{A.57}$$

Then, the centroid of the points in position 2 coincide with the centroid of the points obtained by the optimal transformation:

$$\vec{p'} = \vec{p''} = \tilde{R}\vec{p} + \vec{\tilde{T}} \tag{A.58}$$

Proof. Let the error vector $\vec{\epsilon}_i(R,\vec{T})$ be constituted of an average component $\vec{\mu}(R,\vec{T})$ and a varying component with a zero average $\vec{\delta}_i(R,\vec{T})$:

$$\epsilon_i(R,\vec{T}) = \vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T}) \tag{A.59}$$

$$\sum_{i=0}^{n-1}\vec{\delta}_i(R,\vec{T}) = \vec{0} \tag{A.60}$$

Inserting (A.59) into (A.52):

$$\vec{p'}_i = R\vec{p}_i + \vec{T} + \vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T}) \tag{A.61}$$

$$\begin{aligned}
\sigma^2(R,\vec{T}) &\overset{(A.53)}{=} \sum_{i=0}^{n-1} \parallel \vec{p'}_i - R\vec{p}_i - \vec{T} \parallel^2 \\
&\overset{(A.61)}{=} \sum_{i=0}^{n-1} \parallel R\vec{p}_i + \vec{T} + \vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T}) - R\vec{p}_i - \vec{T} \parallel^2 \\
&= \sum_{i=0}^{n-1} \parallel \vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T}) \parallel^2 \\
&= \sum_{i=0}^{n-1} (\vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T}))^T (\vec{\mu}(R,\vec{T}) + \vec{\delta}_i(R,\vec{T})) \\
&= \sum_{i=0}^{n-1} (\parallel \vec{\mu}(R,\vec{T}) \parallel^2 + 2\vec{\mu}^T(R,\vec{T})\vec{\delta}_i(R,\vec{T}) + \parallel \vec{\delta}_i(R,\vec{T}) \parallel^2 \\
&= 2\vec{\mu}^T(R,\vec{T}) \sum_{i=0}^{n-1} \vec{\delta}_i(R,\vec{T}) + \sum_{i=0}^{n-1} (\parallel \vec{\mu}(R,\vec{T}) \parallel^2 + \parallel \vec{\delta}_i(R,\vec{T}) \parallel^2) \\
&\overset{(A.60)}{=} n \parallel \vec{\mu}(R,\vec{T}) \parallel^2 + \sum_{i=0}^{n-1} \parallel \vec{\delta}_i(R,\vec{T}) \parallel^2
\end{aligned} \tag{A.62}$$

There is a gain in forcing $\vec{\mu}(R,\vec{T}) = \vec{0}$ when one is trying to minimize $\sigma^2(R,\vec{T})$, as can be seen by inspection of (A.62). Since $\vec{\mu}(R,\vec{T})$ can be absorbed in $\vec{T}$ (cf. (A.61)), for the optimal transformation $\tilde{R}$ and $\vec{\tilde{T}}$, we will have $\vec{\mu}(\tilde{R},\vec{\tilde{T}}) = \vec{0}$.

$$\vec{p'}_i = \tilde{R}\vec{p}_i + \vec{\tilde{T}} + \vec{\delta}_i(\tilde{R},\vec{\tilde{T}}) \tag{A.63}$$

$$\sum_{i=0}^{n-1} \vec{\delta}_i(\tilde{R},\vec{\tilde{T}}) = \vec{0} \tag{A.64}$$

$$\begin{aligned}
\vec{p'} &\overset{(A.56)}{=} \frac{1}{n}\sum_{i=0}^{n-1}\vec{p'}_i \overset{(A.63)}{=} \frac{1}{n}\sum_{i=0}^{n-1}(\tilde{R}\vec{p_i}+\vec{\tilde{T}}+\vec{\delta}_i(\tilde{R},\vec{\tilde{T}})) \\
&= \frac{1}{n}\sum_{i=0}^{n-1}(\tilde{R}\vec{p_i}+\vec{\tilde{T}})+\frac{1}{n}\sum_{i=0}^{n-1}\vec{\delta}_i(\tilde{R},\vec{\tilde{T}}) \\
&\overset{(A.54)}{=} \frac{1}{n}\sum_{i=0}^{n-1}\vec{p''}_i+\frac{1}{n}\vec{0} \\
\vec{p'} &\overset{(A.57)}{=} \vec{p''} \overset{(A.54)}{=} \frac{1}{n}\sum_{i=0}^{n-1}(\tilde{R}\vec{p_i}+\vec{\tilde{T}}) \overset{(A.55)}{=} \tilde{R}\vec{p}+\vec{\tilde{T}}
\end{aligned}$$

□

Let us introduce the shifted 3D points $\vec{q_i}$ and $\vec{q'}_i$:

$$\vec{q_i} \equiv \vec{p_i}-\vec{p} \tag{A.65}$$

$$\vec{q'}_i \equiv \vec{p'}_i-\vec{p'} \tag{A.66}$$

$$\begin{aligned}
\sigma^2(R,\vec{T}) &\overset{(A.53)}{=} \sum_{i=0}^{n-1} \parallel \vec{p'}_i - R\vec{p_i} - \vec{T} \parallel^2 \\
&\overset{(A.65),(A.66)}{=} \sum_{i=0}^{n-1} \parallel \vec{q'}_i + \vec{p'} - R(\vec{q_i}+\vec{p}) - \vec{T} \parallel^2 \\
&\overset{(A.58)}{=} \sum_{i=0}^{n-1} \parallel \vec{q'}_i + \tilde{R}\vec{p}+\vec{\tilde{T}} - R\vec{q_i} - R\vec{p} - \vec{T} \parallel^2 \qquad (A.67) \\
\sigma^2_{min}(R,\vec{T}) &= \sigma^2(\tilde{R},\vec{\tilde{T}}) = \sum_{i=0}^{n-1} \parallel \vec{q'}_i - \tilde{R}\vec{q_i} \parallel^2 \qquad (A.68)
\end{aligned}$$

Equation (A.68) shows that $\tilde{R}$ can be isolated independently of $\vec{\tilde{T}}$, which can be computed afterwards from (A.58): $\vec{\tilde{T}} = \vec{p'} - \tilde{R}\vec{p}$. The next step is therefore to compute $\tilde{R}$.

A.3.2 Computation of the Optimal Rotation

Our goal now is to find the rotation matrix $\tilde{R}$ such that the sum of the error vectors squared norms is minimized. Let us develop our expression of the error vectors

squared norms sum:

$$\begin{aligned}
\sigma^2(\tilde{R},\vec{\tilde{T}}) &\overset{(A.68)}{=} \sum_{i=0}^{n-1} \parallel \vec{q'}_i - \tilde{R}\vec{q_i} \parallel^2 \\
&= \sum_{i=0}^{n-1} (\vec{q'}_i - \tilde{R}\vec{q_i})^T (\vec{q'}_i - \tilde{R}\vec{q_i}) \\
&= \sum_{i=0}^{n-1} (\parallel \vec{q'}_i \parallel^2 - \vec{q'}_i^T \tilde{R}\vec{q_i} - \vec{q_i}^T \tilde{R}^T \vec{q'}_i + \vec{q_i}^T \tilde{R}^T \tilde{R}\vec{q_i}) \\
&= \sum_{i=0}^{n-1} (\parallel \vec{q'}_i \parallel^2 - 2\vec{q'}_i^T \tilde{R}\vec{q_i} + \parallel \vec{q_i} \parallel^2) \qquad \text{(A.69)}
\end{aligned}$$

From inspection of (A.69), minimizing $\sigma^2(\tilde{R},\vec{\tilde{T}})$ is equivalent to maximizing

$$G \equiv \sum_{i=0}^{n-1} \vec{q'}_i^T \tilde{R}\vec{q_i} \qquad \text{(A.70)}$$

Let us use identity (B.1) in (A.70):

$$\begin{aligned}
G &= \sum_{i=0}^{n-1} \vec{q'}_i^T \tilde{R}\vec{q_i} \overset{(B.1)}{=} \sum_{i=0}^{n-1} trace(\tilde{R}\vec{q_i}\vec{q'}_i^T) \\
&= trace(\tilde{R}\sum_{i=0}^{n-1} \vec{q_i}\vec{q'}_i^T) = trace(\tilde{R}H) \qquad \text{(A.71)}
\end{aligned}$$

where

$$H \equiv \sum_{i=0}^{n-1} \vec{q_i}\vec{q'}_i^T \qquad \text{(A.72)}$$

We are now searching for the rotation matrix $\tilde{R}$ that maximizes $G = trace(\tilde{R}H)$. Let us define the matrix X, built from U_H and V_H, the orthogonal matrices obtained from SVD of H, i.e. $H = U_H D_H V_H^T$.

$$X \equiv V_H U_H^T \qquad \text{(A.73)}$$

X is an orthogonal matrix.

Proof.

$$X^T X = U_H V_H^T \cdot V_H U_H^T = U_H U_H^T = I \tag{A.74}$$

□

XH is a symmetric matrix.

Proof.

$$\begin{aligned}(XH)^T &= (V_H U_H^T \cdot U_H D_H V_H^T)^T = (V_H D_H V_H^T)^T = V_H D_H^T V_H^T = V_H D_H V_H^T \\ &= XH\end{aligned}$$

□

Since XH is positive definite (i.e. symmetric with eigenvalues ≥ 0), it can be expressed in the form

$$XH = AA^T \tag{A.75}$$

This allows us to apply (B.11) to the matrix XH:

$$trace(XH) \geq trace(RXH) \tag{A.76}$$

where R is an orthogonal matrix. Therefore, there is no rotation matrix that could be premultiplied to XH, such that the trace of the obtained matrix RXH would have a greater trace than XH. Hence, X is the solution we were searching for that maximizes $G = trace(\hat{R}H)$.

$$\tilde{R} \overset{(A.73)}{=} V_H U_H^T \tag{A.77}$$

$$\vec{\tilde{T}} \overset{(A.58)}{=} \vec{p'} - \tilde{R}\vec{p} \tag{A.78}$$

A.3.3 Summary of the 3D Registration Procedure

To summarize the 3D registration procedure, let us restate the algorithm: Given two sets of corresponding 3D points $\{\vec{p}\}$ and $\{\vec{p'}\}$, expressed in the same reference system. We are searching for the optimal rotation $\tilde{R}$ and the optimal translation $\vec{\tilde{T}}$ such that $\vec{p'}_i = R\vec{p}_i + \vec{T} + \vec{\epsilon}_i(R, \vec{T})$ minimizes the square error sum

$$\sigma^2(R, \vec{T}) = \sum_{i=0}^{n-1} \parallel \vec{p'}_i - R\vec{p}_i - \vec{T} \parallel^2$$

1. Compute the centroids of the two sets of points, $\vec{p}$ and $\vec{p'}$
2. Shift the points with their respective centroids:

$$\begin{aligned} \vec{q}_i &= \vec{p}_i - \vec{p} \\ \vec{q'}_i &= \vec{p'}_i - \vec{p'} \end{aligned}$$

3. Compute the matrix H, obtained from the shifted points:

$$H = \sum_{i=0}^{n-1} \vec{q}_i \vec{q'}_i^T$$

4. Perform SVD on H: $H = U_H D_H V_H^T$
5. The optimal rotation is $\tilde{R} = V_H U_H^T$
6. The optimal translation is $\vec{\tilde{T}} = \vec{p'} - \tilde{R}\vec{p}$

A.4 Transformation of a Moving Reference Frame with Respect to a Fixed Scene

Let us suppose a camera is moving around a rigid object which is fixed with respect to a global reference frame. From the camera point of view, it is as if the scene had undergone a rigid motion with respect to its reference frame. With a sufficient number

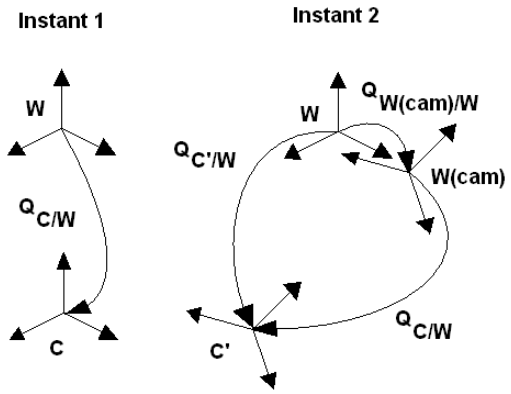


Figure A.4: Registration of two positions of a moving camera

of points in position 1 and in position 2, this transformation can be computed, as shown in Section A.3. We want to use this information to obtain the transformation the camera has gone through between position 1 and position 2.

Figure A.4 shows a camera **C** at instants 1 and 2, along with the global reference frame **W** and the location of the virtual reference frame **W**(**cam**) that has rigidly followed the movement of the camera. The new position $Q_{C'/W}$ of the camera is given by:

$$Q_{C'/W} = Q_{reg} Q_{C/W} \tag{A.79}$$

where

$$Q_{reg} \equiv \begin{bmatrix} \begin{bmatrix} R_{reg} \end{bmatrix} & \begin{bmatrix} T_{reg} \end{bmatrix} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{A.80}$$

Proof. From Section A.3, we could compute the optimal rotation $\tilde{R} = R_{reg}$ and translation $\vec{\tilde{T}} = \vec{T}_{reg}$ that best fit $\vec{X}|_W = R_{reg}\vec{X}|_{W(cam)} + \vec{T}_{reg}$. From the convention of the homogeneous transformation matrices (A.46), we see that the homogeneous transformation relating the $World$ reference frame with the $World(camera)$ reference frames is simply

$$Q_{W(cam)/W} = Q_{reg} \tag{A.81}$$

with the definition of Q_{reg} given in equation (A.80).
Following the transformation arrows of the graph:

$$Q_{C'/W} = Q_{W(cam)} Q_{C/W} \tag{A.82}$$

$$Q_{C'/W} = Q_{reg} Q_{C/W} \tag{A.83}$$

□

Appendix B

Mathematical Identities

This appendix gathers useful mathematical identities that are used in the course of the text, whose derivation is not central in the comprehension of the subject.

B.1 Properties of the Trace of a Matrix

B.1.1 Trace of $B\vec{c}\vec{a}^T$

Let $\vec{a}$ be a $[n \times 1]$ vector, B a $[n \times m]$ matrix and $\vec{c}$ a $[m \times 1]$ vector;

$$\vec{a}^T B\vec{c} = trace(B\vec{c}\vec{a}^T) \tag{B.1}$$

Proof.

$$\begin{aligned}
\vec{a}^T B\vec{c} &= \begin{bmatrix} a_0 & a_1 & ... & a_{n-1} \end{bmatrix} \begin{bmatrix} b_{0,0} & b_{0,1} & ... & b_{0,m-1} \\ b_{1,0} & b_{1,1} & ... & b_{1,m-1} \\ ... & ... & ... & ... \\ b_{n-1,0} & b_{n-1,1} & ... & b_{n-1,m-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ ... \\ c_{m-1} \end{bmatrix} \\
&= \begin{bmatrix} a_0 & a_1 & ... & a_{n-1} \end{bmatrix} \begin{bmatrix} b_{0,0}c_0 + b_{0,1}c_1 + ... + b_{0,m-1}c_{m-1} \\ b_{1,0}c_0 + b_{1,1}c_1 + ... + b_{1,m-1}c_{m-1} \\ ... \\ b_{n-1,0}c_0 + b_{n-1,1}c_1 + ... + b_{n-1,m-1}c_{m-1} \end{bmatrix} \\
&= a_0(b_{0,0}c_0 + b_{0,1}c_1 + ... + b_{0,m-1}c_{m-1}) + a_1(b_{1,0}c_0 + b_{1,1}c_1 + ... + b_{1,m-1}c_{m-1}) \\
&\quad +... + a_{n-1}(b_{n-1,0}c_0 + b_{n-1,1}c_1 + ... + b_{n-1,m-1}c_{m-1}) \qquad \text{(B.2)}
\end{aligned}$$

$$\begin{aligned}
trace(B\vec{c}\vec{a}^T) &= trace\left(\begin{bmatrix} b_{0,0} & b_{0,1} & ... & b_{0,m-1} \\ b_{1,0} & b_{1,1} & ... & b_{1,m-1} \\ ... & ... & ... & ... \\ b_{n-1,0} & b_{n-1,1} & ... & b_{n-1,m-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ ... \\ c_{m-1} \end{bmatrix} \begin{bmatrix} a_0 & a_1 & ... & a_{n-1} \end{bmatrix} \right) \\
&= trace\left(\begin{bmatrix} b_{0,0}c_0 + b_{0,1}c_1 + ... + b_{0,m-1}c_{m-1} \\ b_{1,0}c_0 + b_{1,1}c_1 + ... + b_{1,m-1}c_{m-1} \\ ... \\ b_{n-1,0}c_0 + b_{n-1,1}c_1 + ... + b_{n-1,m-1}c_{m-1} \end{bmatrix} \begin{bmatrix} a_0 & a_1 & ... & a_{n-1} \end{bmatrix} \right) \\
&= a_0(b_{0,0}c_0 + b_{0,1}c_1 + ... + b_{0,m-1}c_{m-1}) + a_1(b_{1,0}c_0 + b_{1,1}c_1 + ... + b_{1,m-1}c_{m-1}) \\
&\quad +... + a_{n-1}(b_{n-1,0}c_0 + b_{n-1,1}c_1 + ... + b_{n-1,m-1}c_{m-1}) \qquad \text{(B.3)}
\end{aligned}$$

Equating (B.2) and (B.3):

$$\vec{a}^T B\vec{c} = trace(B\vec{c}\vec{a}^T)$$

□

B.1.2 Commutability of the Argument of $trace()$

Let A be a $[n \times m]$ matrix and B be a $[m \times n]$ matrix.

$$trace(AB) = trace(BA) \tag{B.4}$$

Proof.

$$\begin{aligned} trace(AB) &= trace\left(\begin{bmatrix} a_{0,0} & a_{0,1} & ... & a_{0,m-1} \\ a_{1,0} & a_{1,1} & ... & a_{1,m-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,1} & ... & a_{n-1,m-1} \end{bmatrix} \begin{bmatrix} b_{0,0} & b_{0,1} & ... & b_{0,n-1} \\ b_{1,0} & b_{1,1} & ... & b_{1,n-1} \\ ... & ... & ... & ... \\ b_{m-1,0} & b_{m-1,1} & ... & b_{m-1,n-1} \end{bmatrix}\right) \\ &= a_{0,0}b_{0,0} + a_{0,1}b_{1,0} + ... + a_{0,m-1}b_{m-1,0} + a_{1,0}b_{0,1} + a_{1,1}b_{1,1} + ... + a_{1,m-1}b_{m-1,1} \\ &\quad + ... + a_{n-1,0}b_{0,n-1} + a_{n-1,1}b_{1,n-1} + ... + a_{n-1,m-1}b_{m-1,n-1} \end{aligned} \tag{B.5}$$

$$\begin{aligned} trace(BA) &= trace\left(\begin{bmatrix} b_{0,0} & b_{0,1} & ... & b_{0,n-1} \\ b_{1,0} & b_{1,1} & ... & b_{1,n-1} \\ ... & ... & ... & ... \\ b_{m-1,0} & b_{m-1,1} & ... & b_{m-1,n-1} \end{bmatrix} \begin{bmatrix} a_{0,0} & a_{0,1} & ... & a_{0,m-1} \\ a_{1,0} & a_{1,1} & ... & a_{1,m-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,1} & ... & a_{n-1,m-1} \end{bmatrix}\right) \\ &= a_{0,0}b_{0,0} + a_{1,0}b_{0,1} + ... + a_{n-1,0}b_{0,n-1} + a_{0,1}b_{1,0} + a_{1,1}b_{1,1} + ... + a_{n-1,1}b_{1,n-1} \\ &\quad + ... + a_{0,m-1}b_{m-1,0} + a_{1,m-1}b_{m-1,1} + ... + a_{n-1,m-1}b_{m-1,n-1} \\ &= a_{0,0}b_{0,0} + a_{0,1}b_{1,0} + ... + a_{0,m-1}b_{m-1,0} + a_{1,0}b_{0,1} + a_{1,1}b_{1,1} + ... + a_{1,m-1}b_{m-1,1} \\ &\quad + ... + a_{n-1,0}b_{0,n-1} + a_{n-1,1}b_{1,n-1} + ... + a_{n-1,m-1}b_{m-1,n-1} \end{aligned} \tag{B.6}$$

Equating (B.5) and (B.6):

$$trace(AB) = trace(BA)$$

□

B.1.3 Trace of a Product of Matrices

Let $\{A_i\}$ be a set of N matrices whose dimensions are such that their product is defined with their neighbors and such that the number of rows of A_1 equals the

number of columns of A_N.

$$trace(A_1 \cdot A_2 \cdot ... \cdot A_N) = trace(A_j \cdot A_{j+1} \cdot ... \cdot A_N \cdot A_1 \cdot A_2 \cdot ... \cdot A_{j-1}) \tag{B.7}$$

Proof.

$$\begin{aligned} trace(A_1 \cdot A_2 \cdot ... \cdot A_N) &= trace(A_1 \cdot (A_2 \cdot ... \cdot A_N)) \\ &\overset{(B.4)}{=} trace(A_2 \cdot A_3 \cdot ... \cdot A_N \cdot A_1) \end{aligned} \tag{B.8}$$

Applying (B.8) $(j-1)$ times:

$$trace(A_1 \cdot A_2 \cdot ... \cdot A_N) = trace(A_j \cdot A_{j+1} \cdot ... \cdot A_N \cdot A_1 \cdot A_2 \cdot ... \cdot A_{j-1})$$

□

B.1.4 Trace of $A^T BA$

Let A and B be two $[n \times n]$ matrices.

$$trace(A^T BA) = \sum_{i=0}^{n-1} \vec{a}_i^T B \vec{a}_i \tag{B.9}$$

where $\vec{a}_i$ is the i^{th} column of A.

Proof.

$$\begin{aligned} A^T BA &= \begin{bmatrix} \vec{a}_0^T \\ \vec{a}_1^T \\ ... \\ \vec{a}_{n-1}^T \end{bmatrix} \begin{bmatrix} B\vec{a}_0 & B\vec{a}_1 & ... & B\vec{a}_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} \vec{a}_0^T B\vec{a}_0 & \vec{a}_0^T B\vec{a}_1 & ... & \vec{a}_0^T B\vec{a}_{n-1} \\ \vec{a}_1^T B\vec{a}_0 & \vec{a}_1^T B\vec{a}_1 & ... & \vec{a}_1^T B\vec{a}_{n-1} \\ ... & ... & ... & ... \\ \vec{a}_{n-1}^T B\vec{a}_0 & \vec{a}_{n-1}^T B\vec{a}_1 & ... & \vec{a}_{n-1}^T B\vec{a}_{n-1} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} trace(A^T BA) &= \vec{a}_0^T B\vec{a}_0 + \vec{a}_1^T B\vec{a}_1 + ... + \vec{a}_{n-1}^T B\vec{a}_{n-1} \\ &= \sum_{i=0}^{n-1} \vec{a}_i^T B\vec{a}_i \end{aligned}$$

□

B.1.5 Schwartz-Cauchy Inequality Applied to Vectors of Dimension n

Let $\vec{a}$ and $\vec{b}$ be two vectors of dimension $[n \times 1]$ whose values are real.

$$(\vec{a}^T\vec{b})^2 \leq (\vec{a}^T\vec{a}) \cdot (\vec{b}^T\vec{b}) \tag{B.10}$$

Proof.

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(a_i b_j - a_j b_i)^2 \geq 0$$

$$\begin{gathered} (a_0^2b_0^2 - 2a_0^2b_0^2 + a_0^2b_0^2) + (a_0^2b_1^2 - 2a_0a_1b_0b_1 + a_1^2b_0^2) \\ +... + (a_0^2b_{n-1}^2 - 2a_0a_{n-1}b_0b_{n-1} + a_{n-1}^2b_0^2) + (a_1^2b_0^2 - 2a_0a_1b_0b_1 + a_0^2b_1^2) \\ +(a_1^2b_1^2 - 2a_1^2b_1^2 + a_1^2b_1^2) + ... + (a_1^2b_{n-1}^2 - 2a_1a_{n-1}b_1b_{n-1} + a_{n-1}^2b_1^2) + ...+ \\ (a_{n-1}^2b_0^2 - 2a_0a_{n-1}b_0b_{n-1} + a_0^2b_{n-1}^2) + (a_{n-1}^2b_1^2 - 2a_1a_{n-1}b_1b_{n-1} + a_1^2b_{n-1}^2) \\ +... + (a_{n-1}^2b_{n-1}^2 - 2a_{n-1}^2b_{n-1}^2 + a_{n-1}^2b_{n-1}^2) \geq 0 \end{gathered}$$

$$\begin{gathered} 2[a_0^2b_0^2 + a_0^2b_1^2 + ... + a_0^2b_{n-1}^2 \\ +a_1^2b_0^2 + a_1^2b_1^2 + ... + a_1^2b_{n-1}^2 \\ +... + a_{n-1}^2b_0^2 + a_{n-1}^2b_1^2 + ... + a_{n-1}^2b_{n-1}^2] \geq \\ 2[a_0^2b_0^2 + 2a_0a_1b_0b_1 + 2a_0a_2b_0b_2 + ... + 2a_0a_{n-1}b_0b_{n-1} \\ +a_1^2b_1^2 + 2a_1a_2b_1b_2 + ... + 2a_1a_{n-1}b_1b_{n-1} \\ +... \\ +a_{n-1}^2b_{n-1}^2] \end{gathered}$$

$$\begin{aligned}(a_0^2 + a_1^2 + ... + a_{n-1}^2)(b_0^2 + b_1^2 + ... + b_{n-1}^2) &\geq (a_0b_0 + a_1b_1 + ... + a_{n-1}b_{n-1})^2 \\ (\vec{a}^T\vec{a}) \cdot (\vec{b}^T\vec{b}) &\geq (\vec{a}^T\vec{b})^2\end{aligned}$$

□

B.1.6 Maximum Trace of RAA^T

Let R be an $[n \times n]$ orthogonal matrix and let A be an $[n \times n]$ matrix.

$$trace(AA^T) \geq trace(RAA^T) \tag{B.11}$$

Proof.

$$trace(RAA^T) \overset{(B.7)}{=} trace(A^TRA) \overset{(B.9)}{=} \sum_{i=0}^{n-1} \vec{a}_i^T R\vec{a}_i \tag{B.12}$$

where $\vec{a}_i$ is the i^{th} column of A. Let us apply the Schwartz-Cauchy inequality (B.10) to the vectors $\vec{a}_i$ and $R\vec{a}_i$:

$$\begin{aligned}(\vec{a}_i^T(R\vec{a}_i))^2 &\leq (\vec{a}_i^T\vec{a}_i) \cdot ((R\vec{a}_i)^T \cdot R\vec{a}_i) \\ \vec{a}_i^T R\vec{a}_i &\leq \sqrt{(\vec{a}_i^T\vec{a}_i)(\vec{a}_i^T R^T R\vec{a}_i)} \\ \vec{a}_i^T R\vec{a}_i &\leq \sqrt{(\vec{a}_i^T\vec{a}_i)(\vec{a}_i^T\vec{a}_i)} \\ \vec{a}_i^T R\vec{a}_i &\leq \vec{a}_i^T\vec{a}_i\end{aligned} \tag{B.13}$$

$$trace(RAA^T) \overset{(B.12)}{=} \sum_{i=0}^{n-1} \vec{a}_i^T R\vec{a}_i$$

$$trace(RAA^T) \overset{(B.13)}{\leq} \sum_{i=0}^{n-1} \vec{a}_i^T\vec{a}_i$$

$$trace(RAA^T) \leq trace(AA^T)$$

□

B.2 Properties of the Determinant of a Matrix

B.2.1 Effect of the Sign Inversion of a Row on the Determinant

Let B be an $n \times n$ matrix built by inverting the sign of one of the rows of the $n \times n$ matrix A. Then,

$$|B| = -|A| \tag{B.14}$$

Proof. Let A be

$$A \equiv \begin{bmatrix} a_{0,0} & a_{0,1} & ... & a_{0,n-1} \\ a_{1,0} & a_{1,1} & ... & a_{1,n-1} \\ ... & ... & ... & ... \\ a_{m,0} & a_{m,1} & ... & a_{m,n-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,1} & ... & a_{n-1,n-1} \end{bmatrix} \tag{B.15}$$

$$|A| = (-1)^m a_{m,0} \begin{vmatrix} a_{0,1} & a_{0,2} & ... & a_{0,n-1} \\ ... & ... & ... & ... \\ a_{m-1,1} & a_{m-1,2} & ... & a_{m-1,n-1} \\ a_{m+1,1} & a_{m+1,2} & ... & a_{m+1,n-1} \\ ... & ... & ... & ... \\ a_{n-1,1} & a_{n-1,2} & ... & a_{n-1,n-1} \end{vmatrix} -(-1)^m a_{m,1} \begin{vmatrix} a_{0,0} & a_{0,2} & ... & a_{0,n-1} \\ ... & ... & ... & ... \\ a_{m-1,0} & a_{m-1,2} & ... & a_{m-1,n-1} \\ a_{m+1,0} & a_{m+1,2} & ... & a_{m+1,n-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,2} & ... & a_{n-1,n-1} \end{vmatrix} \tag{B.16}$$

$$+ - ...$$

$$B \equiv \begin{bmatrix} a_{0,0} & a_{0,1} & ... & a_{0,n-1} \\ a_{1,0} & a_{1,1} & ... & a_{1,n-1} \\ ... & ... & ... & ... \\ -a_{m,0} & -a_{m,1} & ... & -a_{m,n-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,1} & ... & a_{n-1,n-1} \end{bmatrix} \tag{B.17}$$

$$|B| = -(-1)^m a_{m,0} \begin{vmatrix} a_{0,1} & a_{0,2} & ... & a_{0,n-1} \\ ... & ... & ... & ... \\ a_{m-1,1} & a_{m-1,2} & ... & a_{m-1,n-1} \\ a_{m+1,1} & a_{m+1,2} & ... & a_{m+1,n-1} \\ ... & ... & ... & ... \\ a_{n-1,1} & a_{n-1,2} & ... & a_{n-1,n-1} \end{vmatrix} + (-1)^m a_{m,1} \begin{vmatrix} a_{0,0} & a_{0,2} & ... & a_{0,n-1} \\ ... & ... & ... & ... \\ a_{m-1,0} & a_{m-1,2} & ... & a_{m-1,n-1} \\ a_{m+1,0} & a_{m+1,2} & ... & a_{m+1,n-1} \\ ... & ... & ... & ... \\ a_{n-1,0} & a_{n-1,2} & ... & a_{n-1,n-1} \end{vmatrix} \tag{B.18}$$

$$- + ...$$

$$|B| = -|A| \tag{B.19}$$

□

B.2.2 Determinant of $-A$

Let A be an $n \times n$ matrix.

$$|-A| = (-1)^n |A| \tag{B.20}$$

Proof. Applying n times equation (B.14):

$$|-A| = (-1)^n |A| \tag{B.21}$$

□

B.2.3 Product of Eigenvalues

Let A be an $n \times n$ matrix whose eigenvalues are $\lambda_1, \lambda_2, ..., \lambda_n$.

$$\prod_{i=1}^{n} \lambda_i = |A| \tag{B.22}$$

Proof. Let $f(\lambda)$ be the characteristic equation of A:

$$f(\lambda) \equiv |\lambda I - A| = (\lambda - \lambda_1)(\lambda - \lambda_2)...(\lambda - \lambda_n) \tag{B.23}$$

$$f(0) = |-A| = (-\lambda_1)(-\lambda_2)...(-\lambda_n) \tag{B.24}$$

From (B.20):

$$(-1)^n|A| = (-1)^n \prod_{i=1}^{n} \lambda_i \tag{B.25}$$

$$|A| = \prod_{i=1}^{n} \lambda_i \tag{B.26}$$

□

B.3 Properties of Homogeneous Transformation Matrices

B.3.1 Commutability of Matrices Whose Rotation Components Are Small

Let Q_1 and Q_2 be two homogeneous transformation matrices whose roll, pitch and yaw are much smaller than 1. In the first approximation,

$$Q_1 Q_2 = Q_2 Q_1 \tag{B.27}$$

Proof. Let us approximate Q_1 and Q_2 in the first order of small angles, i.e. for a small angle α, $sin(\alpha) \simeq \alpha$ and $cos(\alpha) \simeq 1$:

$$\begin{aligned} Q_1 Q_2 &\simeq \begin{bmatrix} 1 & -\theta_1 & \phi_1 & Tx_1 \\ \theta_1 & 1 & -\psi_1 & Ty_1 \\ -\phi_1 & \psi_1 & 1 & Tz_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\theta_2 & \phi_2 & Tx_2 \\ \theta_2 & 1 & -\psi_2 & Ty_2 \\ -\phi_2 & \psi_2 & 1 & Tz_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\simeq \begin{bmatrix} 1 & -\theta_1 - \theta_2 & \phi_1 + \phi_2 & Tx_1 + Tx_2 \\ \theta_1 + \theta_2 & 1 & -\psi_1 - \psi_2 & Ty_1 + Ty_2 \\ -\phi_1 - \phi_2 & \psi_1 + \psi_2 & 1 & Tz_1 + Tz_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{B.28}$$

Since each entry of (B.28) is symmetrical with respect to the subscripts, $Q_1Q_2 = Q_2Q_1$.

□

B.4 Effect of rounding to the closest integer pixel

Let us assume a 3D point should be imaged at point $\vec{u}_{ideal}$. Due to rounding to the closest integer pixel, its image will be detected as $\vec{u}_{rounded}$. The net difference between these two vectors is a random vector whose values in the $U-$ and $V-$ directions are random variables, uniformly distributed between -0.5 and +0.5. The standard deviation between the ideal and the rounded feature point will be:

$$\sigma = \sqrt{\frac{1}{6}} = 0.408 \tag{B.29}$$

Proof.

$$\begin{aligned}
\sigma^2 &= \int_{-0.5}^{0.5}\int_{-0.5}^{0.5}(u^2+v^2)dudv \\
&= \int_{-0.5}^{0.5}[(\frac{u^3}{3}+uv^2)|_{-0.5}^{0.5}]dv \\
&= \int_{-0.5}^{0.5}[\frac{1}{12}+y^2]dy = (\frac{y}{12}+\frac{y^3}{3})|_{-0.5}^{0.5} \\
&= \frac{1}{6} \\
\sigma &= \sqrt{\frac{1}{6}} = 0.408
\end{aligned} \tag{B.30}$$

□