Camera calibration for the Nasa Virtual GloveBox project

Ramzi Wasef El-Khater

University of Nevada, Las Vegas
NOTE TO USERS

This reproduction is the best copy available.

UMI®
CAMERA CALIBRATION FOR THE NASA
VIRTUAL GLOVEBOX PROJECT

by

Ramzi Wasef El-Khater
Bachelor of Science
University of Nevada, Las Vegas
2002

A thesis submitted in partial fulfillment
of the requirement for the

Master of Science Degree in Computer Science
School of Computer Science
College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2004
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 1427400
Copyright 2005 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Thesis Approval
The Graduate College
University of Nevada, Las Vegas

November 19 _________ 20 04

The Thesis prepared by

Ramzi Wasef El-khater

Entitled

"Camera Calibration for the NASA Virtual GloveBox Project"

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Camera Calibration for the NASA Virtual GloveBox Project

by

Ramzi Wasef El-Khater

Dr. Evangelos A. Yfantis, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In this thesis, we develop a new camera calibration algorithm for use in the NASA Virtual GloveBox (VGX) Human Computer Interaction (HCI) System. The developed algorithm is based on the geometric properties of perspective projections and provides a closed form solution for all the camera parameters. The accuracy of the calibration is evaluated in the context of the NASA Virtual GloveBox and the results indicate that accuracy similar to other more complex and computationally expensive calibration methods can be achieved using our algorithm. Camera calibration is an initial step in many computer vision applications. Computer vision refers to the automatic extraction of information about a scene based on images of it. The task of camera
calibration is to estimate the camera parameters which in turn allow us to infer metric information about a scene from images of it.
LIST OF FIGURES

Figure 1.1  Projection of $a_w$ ............................................................ 6
Figure 2.1  Perspective Projections .................................................. 6
Figure 2.2  Coordinate Reference Frames of Pinhole Camera .......... 8
Figure 2.3a Negative Radial Distortion ......................................... 10
Figure 2.3b Positive Radial Distortion ......................................... 10
Figure 3.1  Configuration for Finding Internal Parameters .......... 19
Figure 3.2  Finding the Focal Length ............................................ 20
Figure 3.3  Finding the Aspect Ratio ............................................ 22
Figure 3.4a Plot of $d_i$ versus .................................................... 24
Figure 3.4b $d(r)$ as a function of $r_i$ ........................................... 24
Figure 3.5a Distorted Image ......................................................... 25
Figure 3.5b Image after Correcting for Radial Distortion ............ 25
Figure 3.6  Measuring the corners of the camera ....................... 26
Figure 3.7  Rotation of the camera by $\theta$ degrees about the x-axis... 30
Figure 3.8  Circular Calibration Pattern ..................................... 33
Figure 4.1  The Grid and Circular Patterns .................................. 36
Figure 4.2  Examples of the Region Selection Tool ...................... 37
Figure 4.3  Regions used for ordering the dots ......................... 38
Figure 4.4  Sorting the Grid Points ............................................. 39
Figure B.1  Rotation about an Arbitrary Axis .............................. 52
ACKNOWLEDGMENTS

I extend my gratitude to my advisor and mentor Dr. Yfantis for giving me the opportunity to work on such an exciting project and for the scientific guidance and contributions throughout the development of this work.

I would also like to thank: My mom and dad without whose love and support none of this would have been possible, my uncle and aunt, Majed and Nadia Khater for all of their generous support throughout my academic career at UNLV. Thanks also to my lovely girlfriend Maria for giving me encouragement and for the numerous discussions and hours spent helping me complete this work.

Many thanks also go out to my friends and colleagues: Oksana Kubushyna, Johnney Istle and Kristal Sauer all of whose work has contributed greatly to this thesis. Finally I would like to thank all my committee members for taking the time to review this work.
CHAPTER 1

INTRODUCTION

Computer vision refers to the automatic extraction of information from a scene based on one or more images of it. When a camera acquires an image of a scene it effectively performs a perspective projection of the scene onto the image plane. The task of camera calibration is to estimate the parameters of the camera so that we are able to perform the reverse perspective projection on the image and estimate the world coordinates of every point. In perspective projection more than one 3D point can be projected onto the same point in the image. Therefore it is impossible to infer the depth (distance from the camera) of a point in the image with only one camera; this is demonstrated in Figure 1.1. In the figure, the point $a_w$ is projected to $a_t$ in the image through the pinhole $p$ and any point along the line $pa_w$ is also projected to $a_t$, making it impossible from looking at $a_t$ to know which point it corresponds to along the line $pa_w$. However, we can use two images of a scene, taken by a pair of calibrated cameras, to resolve the depth of an object.
The camera calibration problem can be formally stated as follows: given the 2D coordinates of a point $a_i$ in the image and a known plane in the world coordinate system, find the 3D world coordinates $a_p$ corresponding to $a_i$ in the plane. In other words, we establish a mapping from the 2D image coordinates to the 3D world coordinates of points on a given plane.

Camera calibration is used in many computer vision applications. Knowing how to map points from the image to world coordinates allows us to extract all kinds of useful information. For instance it can be used to measure distances between certain features on the human face for facial recognition systems. In robot vision a robot can estimate the positions of obstacles automatically using a pair of calibrated cameras. Other applications of camera calibration can be found in the manufacturing industry, where calibrated cameras are used to make
accurate measurements to identify defective units. Camera calibration is also applied in the field of human computer interaction (HCI), the area this thesis deals with.

The camera calibration problem is considered in the context of the NASA Virtual GloveBox (VGX). In this thesis we present a new algorithm for camera calibration based on geometric properties of 3D projections. Chapter 1 gives background information and sets the context in which the calibration problem is considered. Chapter 2 explains in detail the theory of our algorithm. Chapter 3 describes the experimental procedure and presents the experimental results and Chapter 4 gives the conclusive results and future research possibilities. This research was funded by NASA Space Grant/EPSCoR: "Development of a Nationally Competitive Program in Computer Vision Technologies for Effective Human-Computer Interaction in Virtual Environments."
CHAPTER 2

BACKGROUND INFORMATION

2.1 NASA Virtual Glovebox (VGX)

The international space station will soon provide a research facility for studying the long term effects of microgravity on living systems. Experiments will be carried out in a glove box designed to accommodate all the living organisms and necessary instruments. Performing these experiments in the glove box, at microgravity, requires a high degree of skill and training on part of the scientist. The purpose of the Virtual Glovebox (VGX) is to provide scientists with a simulated microgravity environment for training and experimentation on earth [13].

One of the many tasks performed by the VGX is the tracking and understanding of human hand movements [6,7,8]. The VGX should be able to resolve the position of the hand inside the Glovebox and find the position and orientation of every finger. Current VGX designs achieve this by using invasive methods that involve the user wearing magnetically tracked gloves [13]. Instead of using gloves, the VGX at the

4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Computer Graphics and Image Processing Laboratory (CGIPL) at UNLV uses video images of the hands captured from multiple cameras. An initial step in tracking the movement of the hands is calibrating the cameras so that 3D information can be acquired from the 2D images.

The Virtual Glovebox (VGX) system at the CGIPL contains many components which, can be classified into hardware and software. The hardware components include the actual glove box, up to four Pelco CCC1370H-2 Series Digital CCD Cameras with a pixel array of 752x582, one Statistical and Software Analysts Inc. (SSAI) four input video capture card. The SSAI capture card provides real-time (30fps for NTSC) frame grabbing capabilities for up to 4 simultaneous cameras. Thanks to an on-board synchronization circuit, the lag between camera-to-camera switching is minimized. The board includes 4 independent video digitizers for high speed when working with multiple cameras. The design is based on the well-known Video Decoder Fusion878A from Conexant®, along with proprietary switching algorithms and image enhancement capabilities.

The software components of the VGX system include camera calibration, motion detection and hand separation [6][7][8], 3D reconstruction, computer graphics hand model and transmission software. This thesis will deal with the camera calibration component of the VGX system.
2.2 Perspective projections

A perspective projection is a method for projecting a 3D scene onto a 2D plane in a manner that preserves the perspective relations between objects. In a perspective projection objects that are far away from the eye seem smaller than close objects, Figure 2.1 demonstrates how perspective projections work.

In the figure the point \( a_i \) is the projection of the 3D world point \( a \).

We define a 2D coordinate system \( u,v \) in the plane that we are projecting to; this is called the Image plane. The question is, what are the coordinates of \( a_i = (u_i,v_i) \) in this new coordinate system?

![Figure 2.1: Perspective projections](image)

The triangle \( \Delta a_i'pc \) is similar to \( \Delta pa'a' \) therefore,

\[
\frac{u_i}{a_z} = \frac{f}{a_z} \quad \text{or} \quad u_i = -\frac{fa_z}{a_z}
\]  

(2.1)
also $\Delta a_i \parallel p c$ is similar to $\Delta p a^* a^*$, therefore,

$$\frac{v_j}{-a_y} = \frac{f}{a_z} \text{ or } v_j = \frac{-fa_y}{a_z}$$

(2.2)

The point $p$ is called the pinhole; the distance from $p$ to the image plane is called the focal length $f$. The significance of perspective projections lies in the fact that the camera by capturing an image of the 3D world effectively performs a perspective projection on all the points in the scene onto the CCD plane. If we have an ideal camera where the pinhole is positioned along z-axis of the world coordinate system as seen in Figure 2.1, then all we would have to do is perform a reverse perspective projection on the points in the image to obtain their world coordinates. However, the difficulties arise from allowing the camera to be arbitrarily positioned relative to the world coordinate system and from the effects of the cameras internal parameters such as lens distortions and scaling factors on the image.

2.3 The Pinhole Camera Model

The most common method for devising a mathematical model for the camera is the pinhole camera model. The pinhole camera model is inspired from the perspective projection scheme of Figure 2.1, where the pinhole $p$, the focal length $f$ and the image plane together constitute the camera. The pinhole camera model involves four coordinate systems, the world coordinate system (WCS) which is arbitrary, the camera
coordinate system (CCS) which has its origin at the pinhole, the 2D image coordinate system (ICS) which is in pixels and has its origin at the corner of the image, lastly a fourth coordinate system \((u,v)\) is defined with its origin at the center of the image, we will call this the UVCS (uv coordinate system) see Figure 2.2.

\[
\begin{align*}
\text{ICS} & \quad \text{UVCS} \quad \text{CCS} \\
\downarrow & \quad \downarrow & \quad \downarrow \\
c & \quad u & \quad x_c \\
(\epsilon, n) & \quad y & \quad y_c \\
\text{pinhole} & \quad f & \quad z_c \\
x & \quad y & \quad z \\
\text{WCS} & \quad x_w \quad y_w \quad z_w \\
\end{align*}
\]

Figure 2.2 - Coordinate reference frames of pinhole camera model

We start with a point \(p_w = (x_w, y_w, z_w)\) in WCS. To go form the WCS to the CCS we have to rotate the WCS and then translate,

\[
\begin{bmatrix}
x_c \\
y_c \\
z_c 
\end{bmatrix} = R \begin{bmatrix}
x_w \\
y_w \\
z_w 
\end{bmatrix} + \begin{bmatrix}
t_x \\
t_y \\
t_z 
\end{bmatrix},
\]

(2.3)

where \(R\) is the rotation matrix and \(t = [t_x \ t_y \ t_z]\) is the translation vector. Once we have the CCS coordinates we can obtain the UVCS coordinates \((u,v)\) by applying a perspective projection to \((x_c, y_c, z_c)\). We have the following,
\[ u = \frac{-x_c f}{z_c}, \] (2.5)

\[ v = \frac{-y_c f}{z_c}. \] (2.6)

Now that we have \( p_w \) expressed in UVCS, the next step is to express the same point in the ICS. This is achieved by the following,

\[ c - c_0 = s_r u, \] (2.7)

\[ r_0 - r = s_v v, \] (2.8)

where, \((r, c)\) are the image coordinates of a point.

So far in our model, we did not consider any form of lens distortion. However, lens distortion has a significant impact on the shape of the image, especially with inexpensive cameras. The two most commonly considered forms of distortion are the radial and tangential distortions. Radial distortion is due to imperfections in the construction of the lens and causes points to get radially displaced from the center of the image. This displacement can be either positive or negative, as can be seen in Figure 2.3; the square represents the image without distortion and the dashed line represents the distorted image.

Our experiments indicated that the magnitude of distortion increases with the radius. In fact, we can model the magnitude of distortion as a function of the radius using a polynomial of the form

\[ \|d(r_i)\| = k_0 r_i^3 + k_1 r_i^5 \] (2.9)
Figure 2.3 - a). Negative displacement. b). Positive displacement

where \( k_1 \) and \( k_0 \) are the distortion coefficients and \( r_i = \sqrt{u_i^2 + v_i^2} \) is the radius. We estimate the distortion coefficients by projecting a set of test points onto the image plane using the distortion free camera model and fitting a polynomial to the error curve, a more detailed explanation will be given in the next chapter.

Other forms of geometric distortion have been considered in the literature [4]: some of these are the decentering distortion and thin prism distortion. Decentering distortion is caused by the optical centers of the lens elements not being strictly collinear and is governed by the following expressions [4],

\[
\delta_u = p_1(3u^2 + v^2) + 2p_2uv \tag{2.10}
\]
\[
\delta_v = 2p_1uv + p_2(u^2 + 3v^2) \tag{2.11}
\]

where, \( p_1 \) and \( p_2 \) are the distortion parameters and \( \delta_u \) and \( \delta_v \) are the decentering distortions in the u and v directions respectively. Thin prism
distortion arises from imperfections in lens manufacturing and camera assembly and can be expressed as the following functions [4]:

\[ \delta_{wp} = s_1(u^2 + v^2) \]  
\[ \delta_{vp} = s_2(u^2 + v^2) \]

As before, \( s_1 \) and \( s_2 \) are the distortion coefficients and \( \delta_{wp} \) and \( \delta_{vp} \) are the thin prism distortions in the \( u \) and \( v \) directions, respectively. Only radial distortion is considered in this thesis since it is the most significant contributor to distortion in inexpensive cameras.

2.4 Calibration Problem

Now that we have presented the camera model, we can state the calibration problem. The camera calibration problem can be stated as follows: find the 3D world coordinates of a point \( a_p \) on a known plane \( P \) that appears at position \( c \) in the image. In other words, we establish a mapping from the 2D image coordinates to the 3D world coordinates of points on a given plane.

In order to establish this mapping, we need to find the values for all of the parameters of the camera model. The parameters are classified into two classes, internal and external. The internal parameters are the focal length \( f \), the principal point \( (c_o, r_o) \), the scale factors \( s_x \) and \( s_y \) and the distortion parameters. The external parameters are the rotation matrix \( R \) and the translation vector \( t \) in equation (2.7). Once we have
computed the values for all of the parameters, the WCS coordinates of any point in the image can be computed by using equations (2.3) to (2.8).

Thus we obtain

\[
c = \frac{f_x (r_{11} x_w + r_{12} y_w + r_{13} z_w + t_x)}{(r_{51} x_w + r_{52} y_w + r_{53} z_w + t_z)} + c_0, \tag{2.14}
\]

\[
r = r_0 - \frac{f_y (r_{21} x_w + r_{22} y_w + r_{23} z_w + t_y)}{(r_{51} x_w + r_{52} y_w + r_{53} z_w + t_z)}. \tag{2.15}
\]

Using matrix notation, equations (2.14) and (2.15) can be written as,

\[
c = \frac{f_s (R \cdot p_w) + f_t}{R \cdot p_w + t}, \tag{2.16}
\]

\[
r = r_0 - \frac{f_s (R \cdot p_w) + f_t}{R \cdot p_w + t}. \tag{2.17}
\]

where \( f_x = f_s \), \( f_y = f_s \) and \( p_w = [x_w, y_w, z_w] \).

A set of points with known WCS coordinates \((x_w, y_w, z_w)\) is used to solve for the parameters; these points are called “control points”. Once an image of the control points is acquired, the ICS coordinates \((r, c)\) can be extracted from the image. The camera parameters are then solved for by setting a map of correspondences between the known WCS coordinates and the extracted ICS coordinates.

2.5 Literature Survey

The problem of camera calibration has been studied extensively in the literature. In this section we will present some of the techniques and ideas most commonly used. It should be noted that the calibration
algorithm presented later in Chapter 3 is very different than most existing methods. In general, existing calibration methods can be classified into three classes [4].

1. **Closed form solution methods.** In these methods, the non-linear camera calibration problem is transformed into a linear one, the parameters for the linear model are resolved via Linear Least Squares (LLS) (Appendix A) and then the computed parameters are inverse transformed to their non-linear counter parts. The problem with these methods is that it is very difficult to incorporate lens distortions into the model and the accuracy is relatively poor. The method of Abdel-Aziz and Karara [1] falls into this category.

2. **Direct optimization methods.** The parameters of the non-linear model are iteratively estimated via optimization of some error function that relates the known WCS coordinates of the control points to their computed values. Non-linear lens distortion can easily be incorporated into this class. However, the convergence of the non-linear optimization is not guaranteed and usually requires an initial guess that is close to the actual solution. Some of the approaches that fall into this category are those of Brown [2], Faig [14] and Wong [5].

3. **Two-step methods.** This class of methods combines the first two classes together. Non-linear optimization is used to solve for the
parameters. However a closed form solution is used to find the initial guess for the non-linear optimization making these methods more likely to converge on the correct solution. Some the algorithms that fall under this category are those of Faugras and Tosacni [9,10] Weng et al[4], Tsai [11,12], Qing et al [15] and Zhang [16].

One of the first approaches for calibration came from the photogrammetry community [1]. This method is called the Direct Linear Transformation (DLT) and falls in the category of closed form solution methods. The non-linear model of (2.16) and (2.17) is transformed into a linear model which is then solved via LLS. By unifying the denominators of (2.16) and (2.17) we obtain

\[
C = \frac{f_x R_1 p_w + c_0 R_3 p_w + f_x t_x + c_0 t_z}{R_3 p_w + t_z}, \tag{2.18}
\]

\[
r = -\frac{-f_y R_3 p_w + r_0 R_3 p_w - f_y t_y + t_z t_0}{R_3 p_w + t_z}. \tag{2.19}
\]

The following intermediate variables are defined to make the problem linear:

\[
W_1 = f_x R_1 + c_0 R_3
\]

\[
W_2 = r_0 R_3 - f_y R_2
\]

\[
W_3 = R_3
\]

\[
w_4 = f_x t_x + c_0 t_z
\]

\[
w_5 = -f_y t_y + t_z t_0
\]

\[
w_6 = t_z
\]

By considering equations (2.20), (2.18) and (2.19) we obtain,
\[ c = \frac{W_1 p_x + w_4}{W_3 p_x + w_6} \]  
\[ r = \frac{W_2 p_x + w_5}{W_3 p_x + w_6} \]

Since equations (2.21) and (2.22) are linear we can solve for the intermediate parameters \((W_1, W_2, W_3, w_4, w_5, w_6)\) using LLS. To avoid the homogeneous solution, two constraints are set on the parameters:

1. The norm of vector \(W_3\) must equal unity since it is the third row of a rotation matrix.

2. The sign of \(w_6\) must be compatible with the ground truth values; that is \(w_6\) must be positive or negative depending on whether the camera is in front of or behind the image plane.

Once the intermediate values are computed they can be used to establish a mapping from the ICS to the WCS or they can be used to compute the actual camera parameters. The DLT is a simple and computationally efficient method for calibration, however its accuracy is relatively poor when using inexpensive cameras due to the fact that it does not take into consideration lens distortions.

A calibration method that considers a number of different types of lens distortions is the non-linear method developed by Weng et al [4]. This method falls into the two-step methods category. In the first step the DLT method is used to approximate the distortion-free camera parameters. The results from this step are taken as an initial guess for
the non-linear optimization. In the second step all distortions are added to the model and non-linear optimization is used to compute the camera and distortion parameters.

Weng et al employ a very extensive distortion model using three types of distortions. These are the radial, de-centering and thin-prism distortions described in Section 2.3. The total distortion is given by

\[
\delta_u(u,v) = s_i(u^2+v^2)+3p_1u^2+p_2v^2+2p_3uv+k_1u(u^2+v^2)
\]

\[
\delta_v(u,v) = s_2(u^2+v^2)+2p_1uv+p_3u^2+3p_2v^2+k_2v(u^2+v^2)
\]

where \( \delta_u(u,v) \) and \( \delta_v(u,v) \) are the distortion in the u and v directions respectively. Rewriting equations (2.23) and (2.24) we obtain

\[
\delta_u(u,v) = (g_1 + g_3)u^2 + g_4uv + g_7v^2 + k_1u(u^2+v^2)
\]

\[
\delta_v(u,v) = g_2u^2 + g_3uv + (g_2 + g_4)v^2 + k_2v(u^2+v^2)
\]

where

\[
g_1 = s_1 + p_1.
\]

\[
g_2 = s_2 + p_2.
\]

\[
g_3 = 2p_1.
\]

\[
g_4 = 2p_2.
\]

Let \( m=(r_0,c_0,f,t,R) \) be the vector of all camera parameters other than distortion and \( d=(k_1,g_1,g_2,g_3,g_4) \) be the distortion parameters. The first step is to assume zero for all distortion parameters and solve for the remaining parameters using the DLT as described previously. The next
step is to carry out the non-linear optimization. Let \( r_{i}(m,d) \) and \( c_{i}(m,d) \) be the ICS coordinates of a control point computed using the current values for the camera parameters and let \( r \) and \( c \) be the actual ICS coordinates. The non-linear optimization minimizes the following objective function

\[
\sum_{i=1}^{N} \left[ (r_i^r - r_{i}(m,d))^2 + (c_i^r - c_{i}(m,d))^2 \right],
\]

where \( N \) is the number of control points.

Other aspects of calibration play an important role in the overall accuracy as well. Some of these were considered in Heikkila et al [3]: they consider the effects of quantization noise, calibration object accuracy and illumination. The quantization noise is caused by numerous factors, its effects can be reduced by using large dots with large spacing between them. The calibration object should be constructed very precisely in order to reduce any measurement errors, often the errors caused by the calibration object are systematic and can be compensated for. Illumination is one of the most important considerations in any computer vision application and one should always make sure that the calibration object is properly illuminated.
CHAPTER 3

CALIBRATION ALGORITHM

3.1 Finding The Internal Camera Parameters

The internal parameters of the camera are the focal length $f$, the principal point $(c_0, r_0)$ (See Figure 2.2), the scale factors in the $x$ and $y$ direction $S_x$ and $S_y$, and the radial distortion coefficients $k_0$ and $k_1$. Repositioning the camera does not change any of its internal parameters. The only way to change an internal camera parameter is by changing the zoom or the focus setting of the camera which changes the focal length $f$. Therefore, once we have the values of the internal camera parameters, we can reposition the camera as desired.

In order to calculate the internal parameters, we place our calibration object perpendicular to the camera, that is, parallel to the image plane, and we set the origin of the WCS to be on the point that appears at the center of the image. In effect, what we are doing is setting the rotation matrix $R$ to a 180 degree $y$-roll and the translation vector $\tilde{t}$ to $[0 \ 0 \ h]$ where $h$ is the distance from the pinhole to the origin of the WCS (Figure 3.1).
Figure 3.1 - Configuration for finding the internal camera parameters.

3.1.1 The focal length and scale factors

Our algorithm for finding the focal length $f$ relies on the fact that the ratio between the length of a line in real world and the length of that line in the image remains constant for all lines (see Figure 3.2). In the figure we consider the projection of the two lines $x_1x_2$ and $x_3x_4$ where the former is raised by a known height $z_3$. Thus, we have the following equations:

$$\Delta px_1x_2 = \Delta pu_1u_2,$$

so that,

$$\frac{s_x|u_2-u_1|}{|x_2-x_1|} = \frac{f}{h} \quad (3.1)$$

and

$$\Delta px_3x_4 = \Delta pu_3u_4.$$
Therefore,

\[
\frac{s_x |u_4 - u_3|}{|x_4 - x_3|} = \frac{f}{h - z_3} \tag{3.2}
\]

Figure 3.2 - Finding the focal length from two lines

There are three unknowns in equations (3.1) and (3.2). These are: \(s_x\), \(f\) and \(h\). However, if we rearrange equation (3.2) to be

\[
\frac{f}{s_x} = \frac{h |u_2 - u_1|}{|x_2 - x_1|}, \tag{3.3}
\]

we see that there is a constant ratio between \(f\) and \(s_x\). The ratio is constant because both parameters have the same scaling affect on the image. Taking this into consideration, we can define a "Virtual Camera" that has \(s_x\) set to one pixel per centimeter and the corresponding focal
length \( f \) will be much greater than the actual focal length. Thus we eliminate \( s_x \) from equations (3.1) and (3.2). Solving the two equations for \( h \) and \( f \) after setting \( s_x = 1.0 \text{pixel/cm} \) we obtain

\[
h = \frac{Az_3}{A-1},
\]

where

\[
A = \left| \frac{x_2 - x_1}{u_4 - u_3} \right| \left| \frac{x_4 - x_3}{u_2 - u_1} \right|
\]

and

\[
f = \frac{h|u_2 - u_1|}{|x_2 - x_1|}.
\]

Equation (3.6) can be used to calculate \( f \). However, when using equation (3.6) we must consider the effects of radial distortion. Radial distortion causes the length of the lines in the image to be distorted, which leads to an inaccurate estimation of \( f \). To overcome this problem we only consider lines that lie entirely within a certain region in the center of the image where the effects of radial distortion are very small.

Now we proceed to estimate \( s_y \) which is the scaling factor in the \( y \) direction. In many cameras \( s_x \) and \( s_y \) are not equal causing the pixels to have a rectangular shape. The ratio \( \frac{s_x}{s_y} \) is the aspect ratio and will be
denoted by $a$. Figure 3.3 illustrates how we can calculate $a$ using the following equation:

$$a = \frac{s_y}{s_x} = \frac{|y_2 - y_1||u_2 - u_1|}{|x_2 - x_1||v_2 - v_1|}$$  \hspace{1cm} (3.71)$$

![Image plane](image.png)

Figure 3.3 - Finding the aspect ratio from the projection of a horizontal and a vertical line

3.1.2 Estimating Radial Distortion

As it was mentioned earlier, $f$, $s_x$ and $s_y$ were computed using only control points that were close to the center of the image where the effects of radial distortion are negligible. After that, we can use their values to project all the control points from world to image coordinates and use the difference between the projected UVCS coordinates and the observed UVCS coordinates to estimate the radial distortion coefficients. In order
to obtain a good estimate of the distortion the control points used should be uniformly scattered over the entire image.

Let \( \hat{p}_i = (\hat{u}_i, \hat{v}_i) \) be the observed coordinates of a test point in the UVCS and let \( p_i = (u_i, v_i) \) be the distortion-free coordinates corresponding to \( \hat{p}_i \). Using the configuration shown in Figure 3.1, \( p_i \) can be computed from the test point in WCS by projecting it onto the image plane. Thus,

\[
\begin{align*}
    u_i &= -\frac{f x_w}{h}, \\
    v_i &= -\frac{f y_w}{h}
\end{align*}
\]  

(3.8)

The distortion vector \( d_i = \langle d_u, d_v \rangle \) at \( \hat{p}_i \) is given by

\[
d_i = \hat{p}_i - p_i
\]

(3.9)

and the radius \( r_i \) of a point \( \hat{p}_i \) is given by

\[
r_i = \sqrt{\hat{u}_i^2 + \hat{v}_i^2}.
\]

(3.10)

Plotting \( \|d_i\| \) as a function of the radius \( r_i \) shows that the magnitude of the distortion increases with the radius (Figure 3.4 (a)). The magnitude of radial distortion is expressed by a polynomial of the following form [4]

\[
\|d(r_i)\| = k_0 r_i^3 + k_1 r_i^5.
\]  

(3.11)

The coefficients, \( k_0 \) and \( k_1 \) are computed using a least squares fit of \( \|d(r_i)\| \) to \( \|d_i\| \) (See Appendix A). Figure 3.4 (b) shows the plot of \( \|d(r_i)\| \) using the computed values for \( k_0 \) and \( k_1 \). Once the values of \( k_0 \) and \( k_1 \) have been computed, we can calculate the amount of distortion at any given point based on its radius.
Figure 3.4 - a). Plot of $\|d_i\|$ versus $r_i$  

b). $\|d(r_i)\|$ as a function of $r_i$.

Since the direction of the distortion vector lies, by definition, along the radius, we can compute its $d_u$ and $d_v$ components from

$$d_u(r_i) = \|d(r_i)\| \frac{u_i}{r_i}.$$  \hfill (3.12)  

$$d_v(r_i) = \|d(r_i)\| \frac{v_i}{r_i}.$$  \hfill (3.13)

Moreover, we can undistort every point in the image $\hat{p}_i = (\hat{u}_i, \hat{v}_i)$ by using

$$u_i = \hat{u}_i - d_u(r_i).$$  \hfill (3.14)  

$$v_i = \hat{v}_i - d_v(r_i).$$  \hfill (3.15)

Figure 3.5 shows an example of distorted and un-distorted images.
3.2 Physical Camera Calibration Algorithm

After finding the internal camera parameters from Section 3.1, we position the camera at the desired location and proceed to find the translation and rotation.

Two methods have been developed for finding the extrinsic camera parameters: a physical method and an image based one. In the physical calibration method the location and orientation of the camera with respect to the WCS are measured physically, whereas in the image based method the rotation and translation information are inferred from images of the calibration object.

In the physical method the orientation of the camera is obtained by measuring the corners of the box that defines the camera. We measure the coordinates of $p_0$ through $p_7$ as shown in Figure 3.6.
Once we have the corners of the camera we can find the normal to the image plane, \( \mathbf{n} \), by considering the vector from the center of the back plane to the center of the front plane. If we let the center of the back plane be denoted by \( \mathbf{c}_0 = (c_{0x}, c_{0y}, c_{0z}) \) and the center of the front plane denoted by \( \mathbf{c}_1 = (c_{1x}, c_{1y}, c_{1z}) \), we have

\[
\mathbf{c}_0 = \frac{p_0 + p_1 + p_2 + p_3}{4}, \tag{3.15}
\]

\[
\mathbf{c}_1 = \frac{p_4 + p_5 + p_6 + p_7}{4}, \tag{3.16}
\]

\[
\mathbf{n} = \|\mathbf{c}_0 - \mathbf{c}_1\|. \tag{3.17}
\]

Having \( \mathbf{n} \) and knowing the world coordinates of the point \( p = (p_x, p_y, p_z) \) that projects to the center of the image, we can find the equation of a

\[
\text{Figure 3.6 - Measuring the corners of the camera.}
\]
plane $\Pi_1$ with normal $\vec{n}$ that passes through $p$. This plane is parallel to the image plane and has the following equation,

$$n_x(x - p_x) + n_y(y - p_y) + n_z(z - p_z) = 0 \quad (3.18)$$

Thus, $\Pi_1$ intersects the $z=0$ plane at the line,

$$n_x x + n_y y = p_x + n_y p_y + n_z p_z \quad (3.19)$$

Since this line belongs to a plane that is parallel to the image plane, we can use equation (3.13) with the focal length $f$ known from Section 3.1 to calculate $h$. In this case $h$ represents the distance from the pinhole to $p$. Let $(\alpha, \beta, \gamma)$ denote the world coordinates of the pinhole, we can compute the WCS coordinates of the pinhole as follows,

$$\text{pinhole} = p - h\vec{n}. \quad (3.20)$$

To project a point $q=(u,v)$ from the image, we find the world coordinates of $q=(q_x,q_y,q_z)$ in the image plane and intersect the ray that passes through $q$ and the pinhole, with the $z=0$ plane. In order to find $(q_x,q_y,q_z)$ we find the equation of the image plane in WCS and we define vectors $\overrightarrow{du}$ and $\overrightarrow{dv}$ in the direction of increasing $u$ and increasing $v$, respectively.

The world coordinates of the center of the image plane (that is, the principal point or optical axis) can be obtained by traveling a distance of
units $f$ in the direction $-\vec{n}$ from the pinhole. Let $C = (C_x, C_y, C_z)$ be the coordinates of the image plane center in WCS, we have that,

$$C = \text{pinhole} - fn$$

The equation of the image plane given from $C$ and $\vec{n}$ is

$$n_x(x - C_x) + n_y(y - C_y) + n_z(z - C_z) = 0$$

In order to compute $du$ and $dv$ we compute the 3D world coordinates of a point $p_h$ that projects onto the horizontal image axis as well as the coordinates of a point $p_v$ that projects onto the vertical image axis. Once we have the world coordinates of $p_h$ and $p_v$ we project them back onto the image plane through the pinhole, yielding the 3D world coordinates of points along $\overline{du}$ and $\overline{dv}$, respectively. Subtracting $C$ from these points and normalizing the results produces $\overline{du}$ and $\overline{dv}$.

The world coordinates of any point $q = (u,v)$ on the image plane can be obtained from its image coordinates using the following equation,

$$q = C + (u\overline{du} + v\overline{dv})$$

Having the world coordinates of any point $q$ in the image plane we can project $q$ through the pinhole to find its corresponding point on the image plane. This is achieved by intersecting the ray that goes through $q$ and the pinhole with the known plane. The equation of the ray is given by

28
\[ \vec{r}(t) = (1-t)q + t \cdot \text{pinhole}. \] (3.24)

If, for instance, we know that \( q_w \) lies on the \( z=0 \) plane, we can find its coordinates by finding the intersection of \( \vec{r}(t) \) with the plane \( z=0 \).

As mentioned earlier, the extrinsic camera parameters are the rotation and the translation. The rotation matrix can be obtained from \( \vec{n}, \vec{du} \) and \( \vec{dv} \) by

\[
R = \begin{bmatrix} \vec{du} & \vec{dv} & \vec{n} \end{bmatrix}.
\] (3.25)

The translation vector is obtained from the coordinates of the point that projects to the center of the image. Hence, if that point has coordinates \( p = (p_x, p_y, p_z) \) it follows that

\[
\vec{i} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}
\] (3.26)

3.3 Image Based Camera Calibration Algorithm

In the physical calibration method the rotation of the camera is resolved by physically measuring the corners of the camera. This process, however, can be tedious and requires highly accurate and expensive tools to be carried out successfully. In order to overcome these difficulties an implicit calibration method was developed. The implicit method eliminates the need to physically measure the corners of the camera by using a special calibration pattern that allows us to infer the
orientation of the camera from the images. Since, in the NASA GloveBox we are free to choose the WCS origin, we chose it so that it appears at the center of the image. This assumption will be made for the remainder of this section.

Let us assume that the camera is rotated only about the x-axis by $\theta$. That is, the rotation matrix is an x-roll (Figure 3.8).

![Figure 3.8 - Rotation of the camera by $\theta$ degrees about the x-axis](image)

From Figure 3.8 it follows that

$$\triangle ABF \approx \triangle AED.$$ 

Therefore,

$$\frac{AB}{AD} = \frac{BF}{ED} = \hat{\lambda},$$

(3.27)

and
\[ \Delta PED \approx \Delta PCD. \]

Hence,

\[
\frac{ED}{D_C} = \frac{EP}{PC} = \frac{AP - ED \sin \theta}{PC}. \tag{3.28}
\]

Moreover, since

\[ \Delta PBF \approx \Delta P_{B_1}C. \]

we have

\[
\frac{BF}{CB_1} = \frac{PF}{PC} = \frac{AP + BF \sin \theta}{PC}. \tag{3.29}
\]

From (3.27) we have \( BF = \lambda ED \) hence (3.29) becomes

\[
\frac{\lambda ED}{CB_1} = \frac{AP + \lambda ED \sin \theta}{PC}. \tag{3.30}
\]

Substituting for the known values in equations (3.28) and (3.30), we obtain the following two equations with two unknowns,

\[
\frac{ED}{a} = \frac{f - ED \sin \theta}{h}. \tag{3.31}
\]

\[
\frac{\lambda ED}{a} = \frac{f - \lambda ED \sin \theta}{h}. \tag{3.32}
\]

Solving (3.31) and (3.32) for \( \theta \) we get,

\[
\theta = \sin^{-1} \left( \frac{h(\lambda - 1)}{a(\lambda + 1)} \right). \tag{3.33}
\]

We were able to determine \( \theta \) based on the ratio of the projection of \( D_1C \) to the projection of \( CB_1 \) where \( D_1, C \) and \( B_1 \) are three collinear points on the y-axis and \( C \) is the center of \( B_1D_1 \). Referring back to
Figure 3.8, let $\vec{n}_c$ be the normal to the image plane and $\vec{n}_w = <0,0,1>$ be the normal to the $x_w,y_w$ plane. The angle between $\vec{n}_c$ and $\vec{n}_w$ is $\pi + \theta$. Therefore,

$$\vec{n}_c = R_x(\pi + \theta) \vec{n}_w$$  \hspace{1cm} (3.34)

where $R_x(\theta)$ is a rotation matrix about the x-axis. Thus, we are able to find the normal to the image plane $\vec{n}_c$ for the case when the camera is only rotated about the x-axis.

In the general case, however, the image plane is not rotated just about the x-axis. Instead, we have a sequence of rotations about the $x$, $y$ and $z$ axis. To find the angle between $\vec{n}_c$ and $\vec{n}_w$ we first need to identify an axis of rotation in the $x_w,y_w$ plane. Once we have found this axis we can use equation (3.33) to find the rotation angle. Our method for finding the axis relies on the fact that the image of the axis of rotation remains unchanged as the camera is rotated. In order to utilize this fact, we constructed a special calibration grid that consists of points in a circular pattern. The points are positioned on the circle at every 2 degrees (Figure 3.9).

Due to the fact that the length of the axis does not change as the camera is rotated, we find the opposing pair of points that are equidistant to the center of the circle and take the line given by them to the axis. A detailed algorithm for doing this is given in Section 4.2.2.
Once we have found the axis of rotation, we use the line perpendicular to it in equation (3.33) to determine the angle $\theta$.

Knowing the axis of rotation and the angle $\theta$, we can construct a corresponding rotation matrix about the axis as described in Appendix B, say $R_u(\theta)$. This allows us to compute $\nvec{c}$ from $\nvec{w}$ by

$$\nvec{c} = R_u(\pi + \theta) \nvec{w}.\quad (3.36)$$

Once we have the normal to the image plane, $\nvec{c}$, we can compute the WCS coordinates of the pinhole and the center of the image plane, $C=(C_x,C_y,C_z)$, by ray tracing from the WCS origin in the direction of $-\nvec{c}$ as described in the previous section. Using $\nvec{c}$ and $C$ we can find the equation of the image plane; then, $\overline{du}$ and $\overline{dv}$ can be found by intersecting the rays with the image plane as described in the previous
section (see Figure 3.7). We can construct our rotation matrix $\mathbf{R}$ as in (3.25),

$$\mathbf{R} = \begin{bmatrix} \bar{d}u & \bar{d}v & \bar{n}_c \end{bmatrix} \quad (3.37)$$
CHAPTER 4

EXPERIMENTAL PROCEDURE

This chapter will describe the experimental procedure and present the results obtained from our experiments. Section 4.1 presents the physical and software tools needed for the experiment, Section 4.2 describes the procedure of the experiment and Section 4.3 presents the results.

4.1 Experimental Tools

The experimental tools that we used can be divided into two categories, physical tools and software tools. The physical tools are a circular pattern, a grid pattern and a precisely measured box. The software tools used are a region selection tool, a point extraction tool and tools for rearranging extracted points.

The grid pattern is used in finding the internal parameters of the camera (Section 3.1) and in testing the final calibration as will be seen later. The grid uses the same dots used in the circular pattern, but they are drawn in an 11x14 grid with a spacing of 2cm between neighboring dots. The circular pattern is used to carry out the implicit calibration as
described in Section 3.2 (See Figure 4.1). It consists of 180 dots, evenly spaced on a circle with a 10 cm radius. The dots themselves have a circular shape with each having 2 mm radius.

![Figure 4.1: The grid and circular patterns.](image)

The region selection tool is necessary to select the region in the image where the calibration pattern appears so to limit the search for dots to that region. It works by enclosing the region of interest with a closed polygon so that the point extraction tool only considers points that fall inside this polygon. The polygon is specified by the user. It is simply an ordered sequence of vertices with an edge assumed between every two consecutive vertices. The polygon should enclose the entire calibration pattern as seen in Figure 4.2.

Once the region has been selected, the point extraction tool is used to accurately detect the image coordinates of the dots in the calibration patterns.
The first step in extracting the points is converting the image from RGB to YIQ, then, the image is segmented into black and white regions using a threshold technique, and finally, the center of mass of each of the black regions (dots) is found using depth first search giving us the image coordinates of a dot.

Once we have extracted the image coordinates of the dots, we need to arrange them in a meaningful way; the point ordering tools are used for this. There are two tools for ordering points, a tool for arranging the points from the circular pattern into clockwise order and a tool for ordering the points in the grid line by line. The circle ordering tool works by separating the points into four regions in the manner shown in Figure 4.3, each region is specified by a triangle. The points in Region 1 are sorted by their y coordinates in increasing order, in Region 2 they are sorted by decreasing x, in Region 3 by decreasing y and in Region 4 by increasing x. This gives us a clockwise ordering of the points.
Figure 4.3: Regions used for ordering the dots.

The tool for arranging the grid points sorts the points line by line from left to right as shown in Figure 4.4. The algorithm to do this is outlined below.

**Algorithm:** Sorting the grid points line by line.

**Input:** The four corners of the grid are the input to the algorithm. Let \( \{c_{bl}, c_{ul}, c_{br}, c_{ur}\} \) be the corners, where \( c_{bl} \) is the bottom left corner, \( c_{ul} \) is the top left corner, \( c_{br} \) is the bottom right corner and \( c_{ur} \) is the top right corner. (see Figure 4.4)

**Step 1:** Find the head and tail of every line. This is done by finding the slope \( s_i \) of the line \( c_{bl}c_{ul} \), for all remaining points \( p_i \) we find the slope of the line \( p_i c_{ul} \) or \( p_i c_{bl} \), if this is equal to \( s_i \) then \( p_i \) is considered a head of a line. Similarly using \( c_{br}c_{ur} \) we can find all the tails of the lines.
Let *heads* be the array of all heads and *tails* be the array of all tails. Once we have found *heads* and *tails* we sort each set in increasing order according to the y-coordinate.

**Step 2:** Sort the points on every line. This is done similarly to Step 1, we consider every line formed by joining corresponding points in *heads* and *tails*. That is, we consider every line $l_i$ formed by $heads[i]$ and $tails[i]$, let $s_i$ be the slope of this line. For all the remaining points $p_j$ if the slope of the line $p_j, heads[i]$ or $p_j, tails[i]$ equals $s_i$ then $p_j$ belongs to the line $l_i$. Next we sort all the points in every line $l_i$ in increasing distance from $heads[i]$.

**Step 3:** The sequence of lines $l_1, l_2, l_3, ..., l_n$ gives us the desired ordering of the points.

![Figure 4.4: The arrows indicate the way the points are sorted.](image)
4.2 Experimental Procedure

4.2.1 Internal Calibration

We start by finding the internal camera parameters as described in Section 3.1. The camera is placed perpendicular to the grid as shown in Figure 3.1 and an image of the grid is captured, we extract the 2D image coordinates of this grid using the point extraction tool and store them in the ordered set \textit{LowPnts}. Next, we raise the grid using a precisely measured height, and once again extract the points into the ordered set \textit{HighPnts}.

The sets \textit{LowPnts} and \textit{HighPnts} are re-arranged using the point ordering tool described above and then used with equation (3.5) to compute the focal length $f$. Equations (3.4) and (3.5) are computed using all the horizontal lines and vertical lines obtainable form the grid, the results are averaged to yield the final value for $f$ of the virtual camera.

The next step is to compute the radial distortion coefficients. This is achieved by projecting all the points in \textit{LowPnts} from 3D WCS to the 2D ICS using the values for $f$ and the height of the camera $h$ that were computed in the previous step. The actual image coordinates of the points are the ones that were extracted from the image using the point extraction tool, these are used in equation (3.8) along with the projected image coordinates to compute the distortion at every point. Then a
polynomial of the form (3.10) is fit into the plot of distortions using a least squares fit (see Appendix A2) to yield the values for the radial distortion coefficients $k_0$ and $k_1$.

4.2.2 External Calibration

Once we have the internal camera parameters, we can proceed with the external calibration to determine the rotation and translation of the camera. We place the camera at the desired location, and then we position the circular pattern described in Section 4.1 so that the center of the circle appears at the center of the image. Again, the point extraction tool is used to extract the ICS coordinates of the dots and store them in the array CirclePnts[]. The circle ordering tool is used to order the points in CirclePnts[] in clockwise order starting from an arbitrary point. Once we have the points we use the following algorithm to determine the axis of rotation of the camera.

Algorithm: Determining the axis of rotation from the circle

Step 1: Every two points on opposite sides of the circle are paired together into pairs $(p_i, p_{i+2})$ where $i$ is the pair number and $p_i = (c_i, r_i)$ and $p_{i+2} = (c_{i+2}, r_{i+2})$ are the image coordinates of the two points.

Step 2: For every pair $(p_i, p_{i+2})$ the following ratio is calculated,

$$r_i = \frac{\|p_i - c\|}{\|p_{i+2} - c\|}$$
where \( c = (c, r) \) is the image coordinates of the center of the circle.

**Step 3:** The pair that gives \( r \) closest to 1 is taken to be the axis of rotation.

Let the axis rotation found using the above algorithm be denoted by a vector \( \tilde{v} \), we take the two circle points that fall on the line that is perpendicular to \( \tilde{v} \) and use them for \( D_i \) and \( B_i \) in Figure 9, then we can compute the angle of rotation \( \theta \) about \( \tilde{v} \) using equation (3.33). Once we have \( \theta \) we can compute the normal to the image plane \( \tilde{n} \) using (3.36) and the rotation matrix of the camera is given by equation (3.37).

**4.3 Experimental Results**

A set of known control points was used to verify the correctness of the calibration. The set contains 294 control points, positioned every 2 cm in a 21 by 14 grid pattern. First, an image of the control points is captured from each calibrated camera and the points are projected from the ICS to WCS using the parameters computed from the calibration procedure and equations (2.3) through (2.7). The Euclidean distance between the computed WCS coordinates and the actual known world coordinates is taken as the error measure.

Two types of errors are considered, the calibration error and the reconstruction error. In the calibration error the points are assumed to belong to a known plane say the \( z = 0 \) plane and are projected using only
one camera. Before projecting the points the effects of radial distortion are removed from the ICS coordinates of every point, this is done using equations (3.13) and (3.14). Next the point is projected into CCS using equations (2.5) and (2.6). Then, the world coordinates are computed using (2.7).

To compute the reconstruction error, images from the two cameras are used together to resolve all three coordinates simultaneously. This is achieved by projecting the point from each camera and then using triangulation to resolve its world coordinates. Table 4.1 shows the average calibration and reconstruction errors for 8 calibration trials. The experiment was repeated 5 times per camera in order to minimize the errors due to experimental procedure.

<table>
<thead>
<tr>
<th>Trial No.</th>
<th>ERRORS (cm)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td></td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>1 - Cam 1</td>
<td>0.085</td>
<td>0.155</td>
<td>0.158</td>
<td>0.237</td>
<td></td>
</tr>
<tr>
<td>1 - Cam 2</td>
<td>0.127</td>
<td>0.106</td>
<td>0.065</td>
<td>0.178</td>
<td></td>
</tr>
<tr>
<td>2 - Cam 1</td>
<td>0.071</td>
<td>0.120</td>
<td>0.071</td>
<td>0.156</td>
<td></td>
</tr>
<tr>
<td>2 - Cam 2</td>
<td>0.168</td>
<td>0.214</td>
<td>0.177</td>
<td>0.324</td>
<td></td>
</tr>
<tr>
<td>3 - Cam 1</td>
<td>0.195</td>
<td>0.140</td>
<td>0.141</td>
<td>0.278</td>
<td></td>
</tr>
<tr>
<td>3 - Cam 2</td>
<td>0.100</td>
<td>0.185</td>
<td>0.160</td>
<td>0.264</td>
<td></td>
</tr>
<tr>
<td>4 - Cam 1</td>
<td>0.093</td>
<td>0.246</td>
<td>0.231</td>
<td>0.350</td>
<td></td>
</tr>
<tr>
<td>4 - Cam 2</td>
<td>0.112</td>
<td>0.104</td>
<td>0.042</td>
<td>0.158</td>
<td></td>
</tr>
<tr>
<td>5 - Cam 1</td>
<td>0.107</td>
<td>0.057</td>
<td>0.106</td>
<td>0.161</td>
<td></td>
</tr>
<tr>
<td>5 - Cam 2</td>
<td>0.250</td>
<td>0.179</td>
<td>0.150</td>
<td>0.342</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>0.1308</td>
<td>0.1506</td>
<td>0.1301</td>
<td>0.2448</td>
<td></td>
</tr>
</tbody>
</table>

43
Table 4.1: Average calibration errors of 6 calibration trials

<table>
<thead>
<tr>
<th>Trial No.</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.086</td>
<td>0.285</td>
<td>0.197</td>
<td>0.357</td>
</tr>
<tr>
<td>2</td>
<td>0.145</td>
<td>0.138</td>
<td>0.161</td>
<td>0.257</td>
</tr>
<tr>
<td>3</td>
<td>0.101</td>
<td>0.185</td>
<td>0.512</td>
<td>0.553</td>
</tr>
<tr>
<td>4</td>
<td>0.109</td>
<td>0.169</td>
<td>0.160</td>
<td>0.256</td>
</tr>
<tr>
<td>5</td>
<td>0.194</td>
<td>0.188</td>
<td>0.265</td>
<td>0.378</td>
</tr>
<tr>
<td>Average</td>
<td>0.127</td>
<td>0.193</td>
<td>0.259</td>
<td>0.3602</td>
</tr>
</tbody>
</table>

Table 4.2: Average reconstruction errors resulting from the above calibrations.

As can be seen from the above tables, we have an error of about 2 mm in the 3D coordinate reconstruction. Some of the possible sources of error in our calibration are, the construction of the calibration object, the feature extraction, the triangulation error and lens distortions. The calibration pattern was drawn using AutoCAD and printed on a large sheet of paper using a plotter and then it was placed on a flat board to keep it straight. The distance between to adjacent dots was assumed to be 2.0 cm, however, this might not be true due to some bending or crumpling in the paper. The feature extraction accuracy can be affected by non-uniform illumination, this will cause the detected centroid of the control points to be slightly off the actual center. The third source of error is lens distortion, as mention in previous chapter our calibration
model only considers radial lens distortion since it is the most significant contributor, however, considering other forms of distortion such as tangential and thin prism should improve the results of the calibration. The final source of reconstruction error is the triangulation uncertainty, it is known that the accuracy of triangulation depends on the base length of the cameras. However, in the NASA VGX the cameras must both share a large viewing area to allow free hand movement. So they have to be positioned close to each other.
CHAPTER 5

CONCLUSION

This thesis presented a novel approach to camera calibration based on the geometric properties of the camera. The calibration algorithm has various advantages and disadvantages in comparison to other calibration schemes. For instance, it is computationally fast and algorithmically simple. In fact, our algorithm yields a closed form solution without relying on non-linear optimization methods. However, two separate procedures are needed for computing the camera's intrinsic and extrinsic parameters. Fortunately, once the internal camera parameters have been computed, the camera can be placed in different positions without having to recalibrate the internal parameters.

The accuracy of the calibration method can be further improved by considering other forms of lens distortion such as tangential and thin prism distortion. As a direction for future research one can use the parameters obtained from our calibration method as an initial guess to some of the non-linear optimization methods used in the literature [4][9][10]. It will be interesting to see how this will affect the number of iterations performed by the non-linear optimization.
The calibration method presented in this thesis was successfully used to calibrate the cameras in the NASA Virtual GloveBox. Using the calibrated cameras we were able to extract the positions and directions of the hands in the glove box along with the positions and directions of each individual finger with high accuracy. This will eventually lead to hand gesture recognition.
APPENDIX A

LEAST SQUARES CURVE FITTING

In Least Squares curve fitting, we fit a given polynomial \( f(x) \) with unknown coefficients to a set of observed data points. Let \((x_i, y_i)\) be an observed data point and let \(f(x_i)\) be the value of the polynomial evaluated at that point, we solve for the coefficients of \(f(x)\) that minimize the square error over all the observed data points, that is, we minimize the following sum,

\[
e = \sum_i (y_i - f(x_i))^2
\]  

(A.1)

In our case, for the radial distortion estimation, the polynomial is of the form

\[
f(r_i) = k_0 r_i^3 + k_1 r_i^5.
\]  

(A.2)

where \(r\) is the radius. Thus, the error function becomes,

\[
e = \sum_i (y_i - k_0 r_i^3 - k_1 r_i^5)^2.
\]  

(A.3)

To determine the values of \(k_0\) and \(k_1\) that minimize \(e\) we set the partial derivatives to zero and solve for \(k_0\) and \(k_1\). That is,
\[
\frac{\partial e}{\partial k_0} = \sum_i -2r_i^3(y_i - k_0 r_i^3 - k_1 r_i^5) = 0 \tag{A.4}
\]

\[
\frac{\partial e}{\partial k_1} = \sum_i -2r_i^5(y_i - k_0 r_i^3 - k_1 r_i^5) = 0 \tag{A.5}
\]

the above becomes,

\[
-2\sum_i y_i r_i^3 + 2k_0 \sum_i r_i^6 + 2k_1 \sum_i r_i^8 = 0 \tag{A.6}
\]

and

\[
-2\sum_i y_i r_i^5 + 2k_0 \sum_i r_i^8 + 2k_1 \sum_i r_i^{10} = 0 \tag{A.7}
\]

this, after computing the known values, can be rewritten as,

\[
a_0 + k_0 a_3 + k_1 a_2 = 0 \tag{A.8}
\]

\[
b_0 + k_0 b_3 + k_1 b_2 = 0 \tag{A.9}
\]

where \(a_0\) through \(a_2\), and \(b_0\) through \(b_2\) are known. Equations (A.8) and (A.9) can now be simultaneously solved to determine the values for \(k_0\) and \(k_1\).
APPENDIX B

ROTATION ABOUT AN ARBITRARY AXIS

In this section we present a proof of the fact that a rotation about an arbitrary axis is equivalent to a sequence of rotations about the x, y and z axis. Let \( R_x(\theta) \), \( R_y(\theta) \) and \( R_z(\theta) \) be the rotation matrices about the x, y and z axis, respectively. Then, they are given by

\[
R_x(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix} \quad (B.1)
\]

\[
R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta
\end{bmatrix} \quad (B.2)
\]

\[
R_z(\theta) = \begin{bmatrix}
\sin \theta & \cos \theta & 0 \\
-\cos \theta & \sin \theta & 0 \\
0 & 0 & 1
\end{bmatrix} \quad (B.3)
\]

Now let us consider the rotation about an arbitrary axis defined by the vector \( \mathbf{u} = (u_x, u_y, u_z) \). To perform a rotation about \( \mathbf{u} \) we first rotate the entire coordinate system about the z-axis until \( \mathbf{u} \) is on the xz-plane. Then, we rotate the system about the y-axis until \( \mathbf{u} \) is on the z-axis. Next, we perform the desired rotation of \( \theta \) about the z-axis. Finally, we
apply the inverse of the first two rotations to bring the system back to its original position. The rotation matrix for rotating about $\vec{u}$ shown in Figure B.1 is given by,

$$R_u(\theta) = R_z^{-1}(\alpha)R_y^{-1}(\beta)R_z(\theta)R_y(\beta)R_z(\alpha). \quad (B.4)$$

where,

$$\alpha = \sin^{-1} \frac{u_y}{\sqrt{u_x^2 + u_y^2 + u_z^2}} \quad (B.5)$$

and

$$\beta = \sin^{-1} \frac{u_z}{\sqrt{u_x^2 + u_z^2}} \quad (B.6)$$

Therefore, we can express any rotation about an arbitrary axis as a sequence of five rotations about the $x$, $y$ and $z$ axis.
REFERENCES


VITA

Graduate College
University Of Nevada, Las Vegas

Ramzi Wasef El-khater

Home Address:
P.O. Box 70612
Las Vegas, NV 89170

Degrees:
Bachelor of Science, Computer Science 2002
University of Nevada, Las Vegas

Publications:


Thesis Title: Camera Calibration for The NASA Virtual Glovebox

Thesis Examination Committee:
Chairperson, Evangelos Yfantis, Ph. D.
Committee Member, Nina Cooper, Ph. D.
Committee Member, Laxmi Gewali, Ph. D.
Graduate Faculty Representative, Georg Mauer, Ph. D.