Automatic Multi-Camera Extrinsic Calibration In The Wild

Presented by:
Mauro Giovanni Borrageiro
BRRMAU002
Department of Electrical and Electronics Engineering
University of Cape Town

Prepared for:
Dr. Amir Patel
Department of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics Engineering

November 11, 2020

Key words: Extrinsic Camera Calibration, Calibration Objects, Automatic Calibration

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: .....
M. G. Borrageiro

Date: November 11, 2020

Acknowledgements

So firstly this is not exactly how I'd hoped this would turn out but alas.

To my family, mom, dad I can't thank you enough for the opportunity you have given to me through the experience of university. Thank you for your relentless support and love. To my brother, Roberto, I don't know what I would do without you and thank you for supporting me whenever I need you.

I would like to thank Dr. Patel for his guidance and support through this project, I really appreciated it!

To my friends, Noel, Katie and Keenan (Kerm). I'm so grateful for our friendship and the many memories we have enjoy during our time together at UCT. i looked forward to being friends for a long long time. You made mechatronics worth it.

To all my frisbee and adventure friends, thank you for broadening my view and pushing me to do my best.

Abstract

The overall point of the research project was to implement an automatic calibration technique to determine and refine the extrinsic parameters associated with a camera. The project made use of a 1D calibration wand that was automatically tracked and detected by a convolution neural network(CNN). The extracted points from the CNN were used in a trajectory based optimization algorithm that would taking in 3D estimate points along with some initial extrinsic estimates and optimize all the parameters at once , over the entire trajectory. The trajectory optimization technique was found to produce some relatively comparable result to that of checkerboard calibration but only in the case of 2 cameras. Holistically the projected did not achieve the final goal of calibrating a camera arrange made up of 6 cameras, However, it did show that the technique could be promising given correct formulation of the trajectory optimization problem.

Contents

1 Introduction **1**

1.1 Background to the Study 1

1.2 Objectives of this Study 2

1.3 Scope and Limitations 2

1.4 Report Outline 3

2 Literature Review **4**

2.1 Cameras 4

2.1.1 The Pinhole Camera 4

2.1.2 Projective Geometry 5

2.1.3 Camera Lens Distortion 9

2.2 Camera Calibration 11

2.2.1 Methods for Camera Calibration 11

2.2.2 3D Reference Object calibration 12

2.3 Sparse Bundle Adjustment 17

2.4 Convolution Neural Networks for Feature Detection 17

2.5 Trajectory Based Optimization 20

2.5.1 Defining a Trajectory Optimization Problem 20

2.5.2 System Dynamics 21

2.5.3 Constraints 22

2.5.4 Objective Functions 23

3 Methodology **26**

3.1 Project Development Strategy 26
3.1.1 Problem Statement 26
3.1.2 Project Tasks and Requirements 26
3.2 Design Overview 28
3.3 Cameras 29
3.3.1 Specifications 29
3.3.2 Video Footage 30
3.4 Checkerboard Calibration 32
3.4.1 Intrinsic Calibration 32
3.4.2 Extrinsic Calibration 34
3.4.3 Sparse Bundle Adjustment 36

4 Model Design and Development **38**

4.1 Wand Fabrication 38
4.2 DeepLabCut 39
4.2.1 Setup and Process 39
4.2.2 DeepLabCut Iterations Overview 40
4.3 Trajectory Optimization 42
4.3.1 Wand Pose to Describe the Model 43
4.3.2 Trajectory Parameters and Variables 44
4.3.3 Constraints 46
4.3.4 Objective Function and Initialization 47

5 Results **49**

5.1 DeepLabCut Performance Results 49

5.1.1 White Wand Results 49
5.1.2 Colour Wand Results 51
5.2 Trajectory Optimization Results 51
5.2.1 Results For 2 Cameras 52
5.2.2 Results For 3 Cameras 54
5.2.3 Results For 6 Cameras 55

6 Discussion **56**

7 Conclusions **58**

8 Recommendations for Future Work **59**

A Additional Information **63**

List of Figures

2.1 A simple representation of a pinhole camera showing the projection effects due to the nature of light traveling in straight lines. 5

2.2 An illustration showing how a point, **P**, is projected onto the image plane under the projective camera model[1] 6

2.3 An illustration showing the comparison between scenes with no distortion and two forms of radial distortion [1]. 9

2.4 A comparison of three different possible 2D planar calibration patterns, created using [2] 13

2.5 Three unique 3D calibration objects illustrating the varying levels of complexity in the possible approaches. 14

2.6 Figures showing the variations in calibration objects which are referred to as wands or 1D patterns 16

2.7 DeepLabCut analysing the pose of distinctly different subjects, the coloured markers are user defined and track specific motion or behaviours [3]. 19

2.8 A 3D reconstruction of the data points extracted by DeepLabCut from a 6 camera view to further study the motion of cheetah [3] 19

2.9 A comparison of five common cost functions, highlighting the differing influences of large errors. It is clear that the redescending function limits the effect of large outliers by maintaining a constant maximum cost [4] 25

3.1 Cameras arranged to surround an observed volume for multi-camera footage capture. 30

3.2 Two consecutive frames from video footage showing the use of the light rod to synchronize multiple cameras. 31

3.3 Light rod used to synchronize video footage 32

3.4 A frame captured by the GoPro HERO Session 5 that was used in the intrinsic calibration process. The distortion of the GoPro fisheye lens is apparent as straight lines seem to curve. 32

3.5 A Matlab plot showing the various 3D orientations of the checkerboard during intrinsic calibration with respect to the camera 33

3.6 Distorted overlaid frames from Matlab 33

3.7 OpenCV undistored overlaid frames 34

3.8 Checkerboard 3D point reconstruction following the initial estimate of the extrinsic parameters. Point correspondence between cameras (Camera 1 is in the top left): Red(1&2), Green(2&3), Yellow(3&4), Blue(4&5), Cyan(5&6). 36

3.9 Birds-eye view of the checkerboard 3D point reconstruction following the initial estimate of the extrinsic parameters. 36

3.10 Histogram comparing the reprojection error before and after the sparse bundle adjustment. 37

4.1 Three dimensional illustration of the final calibration wand with the 3 identification markers. 38

4.2 The final iteration of the calibration wand used to train DeepLabCut and in the Trajectory Optimization 39

4.3 First wand iteration correctly labeled, Top(Blue), Middle(Green), Bottom(Red) 41

4.4 Second wand iteration correctly labeled in a 6 camera setting, Top(Blue), Middle(Red), Bottom(Green) 42

4.5 The wand model showing the chosen generalized coordinates and physical dimensions required to describe the remaining markers. 43

5.1 A Histogram showing the residual error from the training data set and fitted Gaussian curve after large outliers had been removed. 49

5.2 First wand iteration correctly labeled, Top(Blue), Middle(Green), Bottom(Red) 50

5.3 First wand iteration incorrectly labeled, The middle is labeled as the top and the top is labeled as the bottom. 50

5.4 A Histogram showing the residual error from the training data set and fitted Gaussian curve after large outliers had been removed. 51

5.5 Visualization of the extrinsic calibration results for a 2 camera case using trajectory optimization 52

5.6 Visualization of the extrinsic calibration results for a 2 camera case using trajectory optimization, Top view 53

5.7 Visualization of the extrinsic calibration results for a 3 camera case using trajectory optimization, the orange circle highlights the expected region for the third camera to appear as the cameras were placed in a straight line in practice. 54

5.8 The resulting 6 camera orientation relative to each other and the wand, given the extrinsic calibration from the trajectory optimization. 55

6.1 This figure illustrates the expected out come of the orientations and positions of the cameras. This was plotted using the checkerboard extrinsic parameters. 57

A.1 Video analyzed using the first iteration of DeepLabCut, The likelihood of the middle marker(green) is distinctly low for the majority of the video. The wand was in frame for the entire video. 63

List of Tables

3.1 Specific camera settings used throughout the project for the GoPro HS5 and Iphone 6S. 29

4.1 Bill of Materials for the Calibration Wand 39

4.2 DeepLabCut CNN iterations over the entire project with the associated data set size, whiteWand refers to the first iteration of the wand and colourWand refers to the final version of the wand. 40

4.3 Example of the data format that DeepLabCut returns for analyzed videos 40

4.4 Description of the parameters defined for the wand model 44

4.5 Trajectory Variables associated with the wand model 45

5.1 DeepLabCut evaluation results returning the reprojection error Means in pixels for all points and only those above the cut-off likelihood, Wand white iteration 49

5.2 DeepLabCut evaluation results returning the reprojection error Means in pixels for all points above the cut-off likelihood, Wand colour iteration . 51

5.3 Rodrigues Angles and Translation vectors for 2 Camera setup, comparing the difference between the Wand and Checkerboard calibration 52

5.4 Rodrigues Angles and Translation vectors for 3 Camera setup, comparing the difference between the Wand and Checkerboard calibration 54

Chapter 1

Introduction

1.1 Background to the Study

The applications of computer vision are becoming more prominent in our every day lives thanks to the advancing rate of technology and development of computing. Computer vision can already be seen in hand held devices for facial recognition [5] and is most definitely in the world of robotics for navigation and analysis [6]. Computer vision refers to a sector of artificial intelligence in which a computer takes in information from outside worlds, analyzes it and attempts to make decisions on what has been identified [7].

Two fundamental parts of computer vision can be identified as follows:

- Taking in optical data from the world around us.
- Analyzing the input data to identify certain features such that a decision can be made.

These two fundamental aspects of computer vision are facilitated through all the various types of cameras and the different sectors of machine learning [7]. However, in most applications of computer visions there needs to be some form of relation between what is "seen" in the optical data and the actual 3D world, in other worlds recovering the geometric information of a viewed scene [8]. An example of this could come from 3D motion capture where a particular observation has been recorded and now metric information needs to be recovered in order to create accurate 3D simulations [4].

One of the most common ways to do this is through camera calibration, where calibration could be thought of as attempting to relate known measurements to observed ones. There are various different techniques and algorithms to calibrate cameras and each have their own set of advantages and downfalls as Remondino [9] explains. Some calibration techniques make us of calibration objects with known dimensions such that it can be extracted and used as reference points [10]. Other use self-calibration algorithms which relates viewed static points in a scene to each [10].

A number of the techniques mentioned for camera calibration are somewhat long and tedious. They are also limited specifically with regards to the arrangement of cameras and how they are set up[10]. On the other hand, there are companies who have produced

commercialized systems that can film large volumes with great accuracy [11]. The problem with these systems is that they are relatively expensive and thus not suitable for small studies.

1.2 Objectives of this Study

The aim of the project is to explore the possible approaches that can used to calibrate the extrinsic parameters of a multi-camera systems in the wild. It involves the following tasks:

1. Explore the literature and the various approaches used to calibrate cameras.
2. Identifying suitable calibration methods for multi-cameras arrangements.
3. Fabricating a calibration object and developing a method to track the calibration object.
4. Developing software to perform the extrinsic calibration.
5. Evaluating the performance of the chosen methods against a standard approach for calibration

1.3 Scope and Limitations

The scope of the project was defined as follows:

1. The calibration object would be designed and fabricated within this project. It was only made from simply materials given the limited access to the university campus during the COVID-19 pandemic
2. Existing software was used to track and extract 2D pixel information about the calibration object
3. The performance of the trajectory optimization based calibration would be compared to that of a standard checkerboard calibration.
4. The trajectory optimization was limited to optimizing the extrinsic parameters and not calculating them.

1.4 Report Outline

The project was broken down into three main task, this included choosing a suitable calibration object for multi-camera extrinsic calibration, identifying a suitable means to track and identify the object and lastly developing a trajectory optimization problem to refine the initial extrinsic parameters. The report is broken down into the following sections. The first section is a literature review which covers associated topics and the relevant theory required for the project. There after the methodology better defines the problem being solved and the direction of the project before going into detail of the intrinsic and extrinsic calibration. The design phases are then documented to give a thorough understand on the design choice and give an overview of the project development. There after, the trajectory base calibration is evaluated and compared to that of the checkerboard calibration. Lastly, the results are discussed and concluded upon before suggesting future development.

Chapter 2

Literature Review

2.1 Cameras

Cameras in the modern world are being used for applications that would not have been dreamt of by their creators. Computer vision technologies based off the optical camera cover applications such as facial recognition to unlock smart phones [5], marker-less motion capture used in reconstructing 3D simulated scenes [12] and the multitude of applications in the robotics industry for navigation and analysis [6]. These incredible applications along with the seemingly trivial act of taking pictures can all be related back to the geometry of the pinhole camera and thus are important aspects to consider for camera calibration.

2.1.1 The Pinhole Camera

Cameras are essentially devices that take in light reflections from the 3D world space and transform it into 2D optical images. Cameras have significantly developed from their early forms which consisted of dark rooms or boxes with a single very small hole on one side. The very small aperture known as the pinhole allows a small amount of reflected light to pass through and onto the back wall which, is covered in a light sensitive material. Light reacts with the light sensitive material and eventually an image is developed [4].

An observer would notice that the image is dimly lit mainly because the light radiant power is limited by the size of the aperture [4]. Along with this, the optical image is flipped and mirrored, since light travels in a straight line from the reflecting object through the aperture resulting in top and bottom flipping, while left and right are mirrored [13], this can be seen in Figure 2.1.

Nowadays, cameras have specially manufactured lenses which allow light to pass through them and by doing so, redirect the light beams to a single point on a digital sensor or photographic film. The aperture of modern lens cameras is adjustable and impacts the optical image in two ways. Firstly, it increases the intensity of the image, resolving the "dimness" issue of the pinhole camera. However, it also affects the depth of field where a larger aperture means less of the image is in focus and depth information is lost, creating a blurry effect [4]. Theoretically every point could be in focus in a pinhole image if the

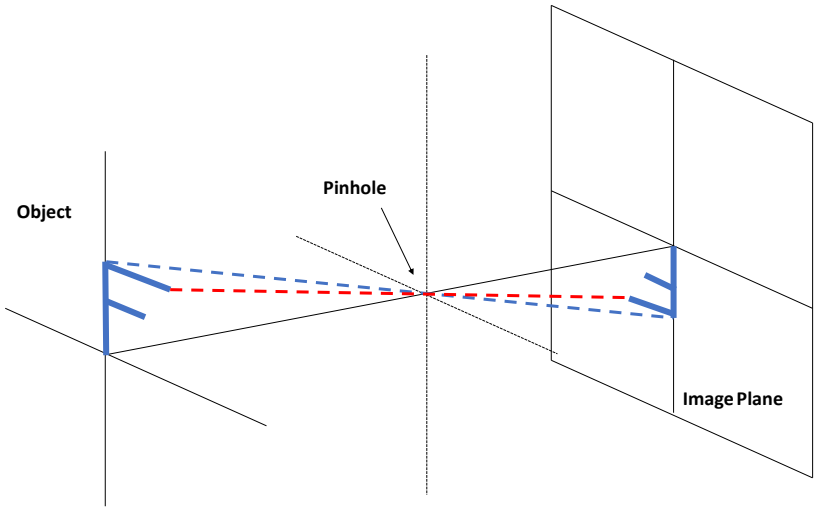


Figure 2.1: A simple representation of a pinhole camera showing the projection effects due to the nature of light traveling in straight lines.

aperture was infinitesimally small but in digital photography, a compromise will be made and desired parts of a scene are chosen to be in focus.

2.1.2 Projective Geometry

The projective model of a camera describes the mathematical relationship between 3D world points and the corresponding 2D points projected onto the image plane. The projective camera model is closely related to the pinhole model spare the addition of distortion coefficients to account for lenses with variable aperture and focal length [14].

In the projective model, the cameras center of reference is aligned with the pinhole location and this is known as the *center of projection* [1]. The **Z**-axis of the camera runs from the center of projection out the aperture in a straight line and is termed the *optical axis*. The image plane of the projective model is situated inbetween the object and the center of projection, in contrast to the pinhole model where it was located after the pinhole. The intersection of the image plane and optical axis is called the *principle point* [4][14]. Lastly, the image plane is located a distance f from the center of projection where f is the *focal length*. The layout of the projective model can be seen below in Figure. 2.2.

An observed 3D point is projected onto the image plane by creating a line that passes through the point and the center of projection. The corresponding 2D point is found at the coordinates of the intersection of said line and the image plane. It is important to note that points on an image are commonly referenced in terms of pixel locations.

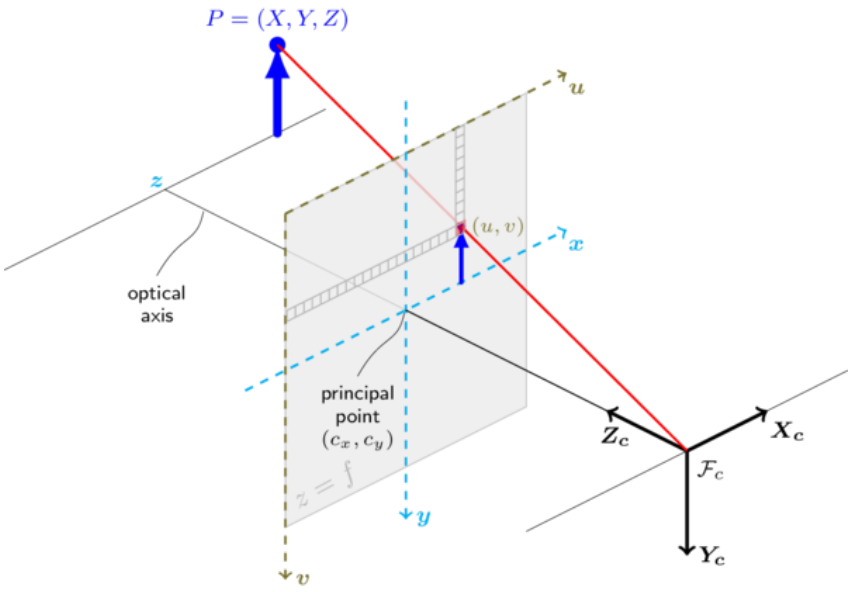


Figure 2.2: An illustration showing how a point, **P**, is projected onto the image plane under the projective camera model[1]

The pixel coordinates are defined as u and v, where the u-axis runs horizontally and the v-axis, vertically. The origin, $(u, v) = (0, 0)$, is taken as the top-left corner of the image.

Determining the Parameters of a Projective Model

The process of determining a camera's projective model starts by looking at the camera's position and orientation in the world frame. The orientation of the camera can be described by a 3×3 rotation matrix, **R**, and the position can be defined by a 3×1 translation vector, **t**. Together, **R** and **t** describe what is known as the *pose* of a camera and further allow a point to be transformed from the world reference frame to the camera reference frame, as shown in equation (2.1):

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t} \tag{2.1}$$

Where subscripts c and w refer to the camera and world reference frame, respectively. Matrices **R** and **t** can be combined to form a joint rotation-translation matrix, $\left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]$.

The joint rotation-translation matrix is facilitated through *homogeneous coordinates*. Homogeneous coordinates are used for their beneficial properties such as simplifying rotation and translation matrices to a single matrix. This is necessary, however, due to the loss of information and properties when transforming points from the Euclidean or Cartesian

space to the projective space. A consequences of transforming from Euclidean to projective space is parallel lines converging to a point, commonly known as a vanishing point in photogrammetry [15]. The homogeneous linear transformation can be seen in equation 2.2

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.2}$$

The pose defines parameters which associate the camera with the external world and are thus known as *extrinsic* parameters. There are of course parameters which are internal to the camera and form part of the *intrinsic* parameters of the camera. The intrinsic parameters are arranged in an upper triangle matrix commonly termed the *camera calibration matrix*, $\mathbf{K}$.

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & \beta & pp_x \\ 0 & f_y & pp_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.3}$$

The intrinsic parameters are utilized in transitioning points from the 3D world to a projective homogeneous image point. A point, $\mathbf{P}$, in the camera world frame is projected onto the image plane, $z = f$, through equations (2.4).

$$\mathbf{P}' = f \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \text{ where } x' = \frac{X_c}{Z_c} \text{ and } y' = \frac{Y_c}{Z_c} \tag{2.4}$$

Where x' and y' are the point coordinates on the image plane $z = f$ and still need to be converted to pixel locations. The x' and y' axes can be seen depicted in blue in Figure. 2.2 above.

The image plane coordinates are then converted to pixel coordinates through scaling x' and y' by the number of pixels on the image sensor. The scaling factors are f_x and f_y, for the respective axes and are made up of the focal length f and an individual pixel scaling term, s_i, thus resulting in $f_x = s_x f$ and similarly so for y'. The conversion also includes an off-set from the principle point since the coordinate origins do not coincide.

$$u = f_x x' + pp_x \quad \textit{and} \quad v = f_y y' + pp_y \tag{2.5}$$

An important aspect of the projective model to note is the scaling factor λ. The reason being that any point on a line that passes through the center of projection will be projected to the *same* point on the image plane. This phenomenon is associated with loss of depth information in transitioning from 3-dimensions to two [4].

Equations (2.5) can be refined to include distortion effects and a *skew* factor, β. The skew factor is often omitted as the effect is negligible [4] [1]. The distortion effect will be covered in the next section but the resulting coordinate conversion is summarized below:

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.6}$$

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & pp_x \\ 0 & f_y & pp_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.7}$$

Conversion of a 3D world point to the camera reference frame:

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & pp_x \\ 0 & f_y & pp_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.8}$$

Point in 3D camera reference frame to the projective space:

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x X_c + pp_x Z_c \\ f_y Y_c + pp_y Z_c \\ Z_c \end{bmatrix} \tag{2.9}$$

Converting to inhomogeneous[1] pixel coordinates:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x X_c + pp_x Z_c}{Z_c} \\ \frac{f_y Y_c + pp_y Z_c}{Z_c} \end{bmatrix} = \begin{bmatrix} f_x x' + pp_x \\ f_y y' + pp_y \end{bmatrix} \tag{2.10}$$

Lastly, the intrinsic and extrinsic parameters are often grouped into a single matrix

[1]The inverse conversion of homogeneous coordinates back to regular coordinates[16]

commonly known as the *Camera Matrix*, $\mathbf{P}$ [14].

$$\mathbf{P} = \mathbf{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \tag{2.11}$$

2.1.3 Camera Lens Distortion

The pinhole model is useful for providing a geometric mapping from the real world to the image plane. However ,cameras with real lenses introduce an effect known as *distortion* which alters the mapping of points to the image plane and in turn, affects the calibration process [1].

Kannala et al. [17] expresses that there are different distortion models for the varying lens forms, such as narrow angle and wide angle lenses. Kannala goes on to explain that the pinhole geometry accompanied with these distortion models can provide a good approximation for conventional cameras. However, it is not strictly the case for cameras with fisheye lenses [17]. A fisheye lens is commonly associated with a very large field of view, approximately 180°with large amounts of radial distortion and requires a more unique model to better account for the associated distortion effect [18].

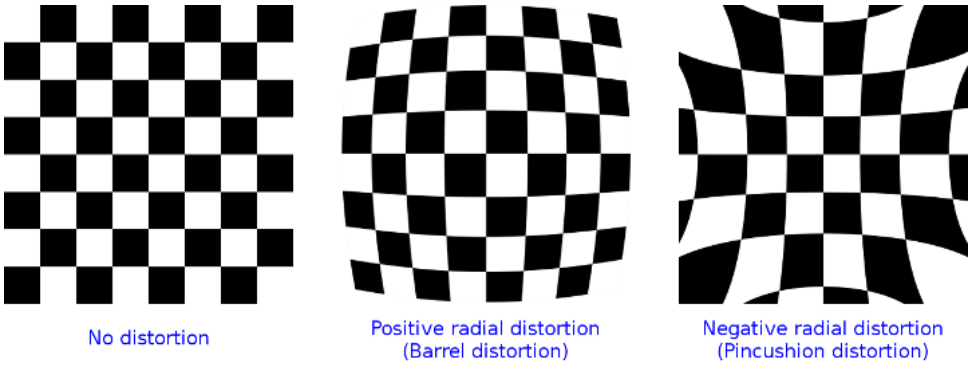


Figure 2.3: An illustration showing the comparison between scenes with no distortion and two forms of radial distortion [1].

There are many forms of distortion but common ones such *radial distortion* and *tangential distortion* are accounted for in open-source computer vision libraries, such as that of OpenCV. The OpenCV library provides distortion models for standard cameras as well as one specifically for cameras with a form of fisheye lens.

Standard Lens Distortion

Lens distortion is introduce to the projective model through a number of coefficients relating to the various forms such as: radial distortion ($k_1..k_6$), tangential distortion

$(p_1..p_2)$ and thin prism distortion $(s_1..s_2)$ [1]. The distortion coefficients are incorporated once a 3D point has been projected to the image plane and the coordinates, x' and y' are known. Note, x' and y' are the same variables described in (2.4). Thus the image plane coordinates which incorporate distortion can be defined as follows:

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^4$$
$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1(r^2 + 2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4$$
$$r^2 = (x')^2 + (y')^2 \tag{2.12}$$

The resulting pixel coordinates, (u, v) become:

$$u = f_x x'' + pp_x \tag{2.13}$$

$$v = f_y y'' + pp_y \tag{2.14}$$

As said by Kannala, the standard camera distortion model is not sufficient for a fisheye camera lens, thus the standard distortion model will be manipulated to account for the very large field of view.

Fisheye Lens Distortion

The necessity for adjusting the standard distortion model when considering fisheye lenses comes from the two models obeying different perspective projections [17]. The fisheye distortion is capable of accounting for the extreme radial distortion that arises from the very large field of view [4]. Such cameras are becoming more popular for computer-vision applications due to their large perspective view [19]. Ontop of this, there availability is becoming more accessible through action cameras, like the GoPro series.

The distortion model used can be described as follows to convert the coordinates on the image plane to consider distortion effects [1][4]:

$$\theta = \arctan(r), \quad r^2 = (x')^2 + (y')^2 \tag{2.15}$$

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{2.16}$$

$$x'' = \frac{\theta_d}{r} x' \quad \text{and} \quad y'' = \frac{\theta_d}{r} y' \tag{2.17}$$

The image plane coordinates are converted to pixels as follows:

$$u = f_x(x'' + \alpha y'') + pp_x, \quad v = f_y y'' + pp_y, \tag{2.18}$$

Lastly the distortion coefficients are group into a matrix **D** and are considered to be part of the *intrinsic* parameters of a camera [14].

$$\mathbf{D} = [k_1, k_2, k_3, k_4] \tag{2.19}$$

The Matrix,**D**, is known as the distortion matrix.

2.2 Camera Calibration

The process of camera calibration is required to obtain geometric information from a 2D scene viewed by a camera [1][18]. This geometric information can then be used in a variety of 3-dimensional computer vision applications, ranging from robotic navigation to 3-dimensional reconstruction and beyond [6].

Its important to firstly define what is meant by camera calibration. Camera calibration implicitly refers to *intrinsic* and *extrinsic* camera parameter calibration. Calibration could be thought of as trying to relate a measurement in the camera reference frame to the same measurement in the world reference frame where we have an understanding of metric information [8]. Davies [8] explains however, that the extent of calibration is dependent on the application in question and the level of accuracy required.

For instance, calibration becomes important when accurate measurements need to be recovered from 2D images. Conversely, in the case of surveillance or counting cars for traffic statistics there may not be a need for accurate measurements or relation between reference frames. This study will place emphasis on achieving a good level of camera calibration.

2.2.1 Methods for Camera Calibration

There are a number of methods through which the intrinsic and extrinsic parameters can be estimated. According to Zhang [10], the techniques associated with camera calibration can somewhat be grouped into two sub-categories: *Self-Calibration* and *3D Reference Object Calibration*. The methods can usually be further broken down into two distinct

steps which involve a linear estimation mainly used for initialization and a second non-linear refinement stage to fine tune the parameter estimations [9].

Reference object calibration refers to the use of a 3-dimensional object whose measurements and geometry are accurate and known. The object is used for point correspondence between images, meaning the 3D object is moved through an observed volume to create a robust array of points that can be used to estimate the intrinsic and extrinsic parameters [10].

In comparison to object based calibration, self-calibration refers to any calibration process which does not employ the use of some 3-dimensional object with known geometry. Self-calibration instead, relies on a camera moving while viewing a rigid scene to create a level of point correspondence. The calibration is then performed by applying constraints to either the parameters of the camera or the image scene itself [9].

Zhang et al. [10][9] provide a comprehensive comparison between the two aforementioned calibration categories, the following section will focus on 3D reference object based calibration and explore the various options within it.

2.2.2 3D Reference Object calibration

With in the section of object based calibration there are a number of techniques which can be employed to achieve point correspondence using diverse object structures and identifying features. These techniques will be grouped into three categories: *3-dimensional*, *2-dimensional* and *1-dimensional* and will be explored further.

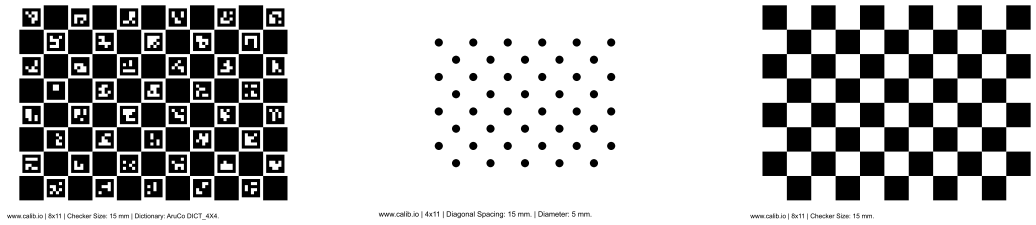
2-Dimensional Objects

2D objects or, commonly referred to as planar objects are exactly that, a flat plane which represents two orthogonal axes with some form of calibration pattern on one of the planar faces. Calibration involving a 2D object is probably the most common technique throughout the computer vision field and is widely referenced [1][10].

It is notably accessible for many applications due to its simplicity of design, essentially composed of a printed checkerboard pattern on a flat planar surface. The checkerboard pattern can be replaced with a number of other pattern options including grid formations of printed dots or unique layouts of ArUco[2] markers [1], different patterns can be seen

[2]An ArUco marker is essentially a binary code embedded in a square marker resembling a quick response (QR) code, See OpenCV's ArUco library for further information.

in Figure 2.4. The important aspects to retain between the variations is the geometric information of the pattern, detailing the position, size and arrangement in the layout.



(a) 8×11 ArUco Marker pattern (b) 8×11 Dot pattern (c) 8×11 Checkerboard Pattern

Figure 2.4: A comparison of three different possible 2D planar calibration patterns, created using [2]

During the calibration process the 2D pattern will be moved through the inspection volume covering a range of orientations to produce a sparse set of points to increase the robustness of the calibration. It is often the corners of the checkerboard that are identified using a corner detector like a Harris filter [4] and thereafter the points correspondence between images is used to find the projective transformation for the camera in question [10].

A notable point on planar calibration is that it can return a large number of corresponding points between images which aids in creating redundancy of data [20]. The reason for this is partly due to the minimum number of points required by calibration algorithms to produce a solution. Once the minimum number of points has been satisfied, any additional data creates redundancies and as a side effect improves the accuracy of the measurement by reducing the effect of noise [21]. Specifically in photogrammetry, increased number of points reduces the effect of noise until a certain point where any additional data does not have a noticeable effect [9].

Some potential pitfalls of planar calibration relate to the intended application. For instance, planar calibration is a relatively straight forward process for applications where all the cameras face in a single direction and the planar object can be viewed by all cameras simultaneously. Fortunately due to the popularity of the method there are software package in the form of OpenCV and Matlab's camera calibration toolboxes which can perform this task [1][22].

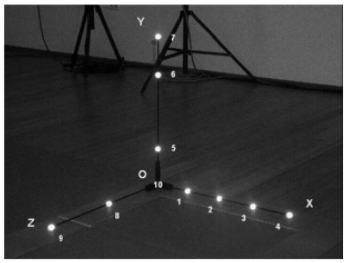
The problem with this method is highlighted when the cameras are positioned in such a way that not all the cameras can see the planar pattern at the same time [18]. This makes scenarios where a subject needs to be surround by cameras or where cameras face each other difficult due to the increased care required to ensure adjacent cameras have an over lapping field of view of the planar pattern for a period in time. The general process to accommodate such cases involves calibrating adjacent cameras pairs which share a common view of planar points at a specific point in time [4][18]. In an example form,

it means finding corresponding points for camera 1 & 2, then 2 & 3, 3 & 4, etc. This process can be made to work however this is not a perfect fix as there needs to be point correspondence between each consecutive camera pair [4].

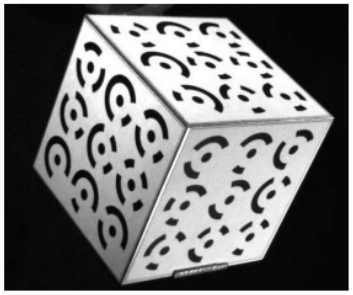
As a whole the planar calibration technique is a reasonable approach due to the relative simplicity in the fabrication process and provides suitable point information to achieve an accurate calibration of the cameras [20]. There are of course pitfalls within the method specifically associated with the limited arrangements and layouts of cameras.

3-Dimensional Objects

3-dimensional calibration objects come in varying arrangements but the common feature amongst all 3D objects is they occupy a volume and can be described with three orthogonal axes. The complexity of the objects also vary and include shapes such as three orthogonal rigid bars with distinctive points to form an orthogonal triad [23], cubes with patterns to identify the unique faces and complex geometric shapes such as a icosahedron, also covered in patterns to identify the unique faces [24].



(a) Orthogonal triad calibration object with distinct points [23]



(b) Cube Calibration Object with identifying patterns [24]

(c) Icosahedron Calibration Object with unique faces [24]

Figure 2.5: Three unique 3D calibration objects illustrating the varying levels of complexity in the possible approaches.

3D calibration objects are less common in comparison to the planar variant and this may be related to the increased effort and precision required in the fabrication process. The 3D objects often require specialist machining to create the object and then a further measurement stage performed by meteorology laboratories to acquire precise 3D positions of the identifying points [20][24]. The identifying points are distinct unique locations on the 3D object that can be located in images and used to determine the intrinsic and extrinsic parameters. Both of these procedures can also be expensive and may be warranted depending on the level of accuracy required [24].

Forbes [24] addresses the requirement of precisely knowing the 3D positions of the iden-

tifying points by offering a calibration procedure that makes use of a 3D object but the identifying points are only approximately known. In Forbes' case the object is covered in 2D coded control points that can be automatically detected to sub-pixel accuracy and in turn returns a good initial approximation [24]. The initial approximation is then optimised via a sparse bundle adjustment to refine the calibration parameters and as a side effect, precise measurements of the 3D calibration are also obtained [24]. Details on the sparse bundle adjustment will be covered in a later section but for context, it is a least-squares optimization procedure commonly used in camera calibration [9]. As a further note, the object used by Forbes still had to be machined for its application.

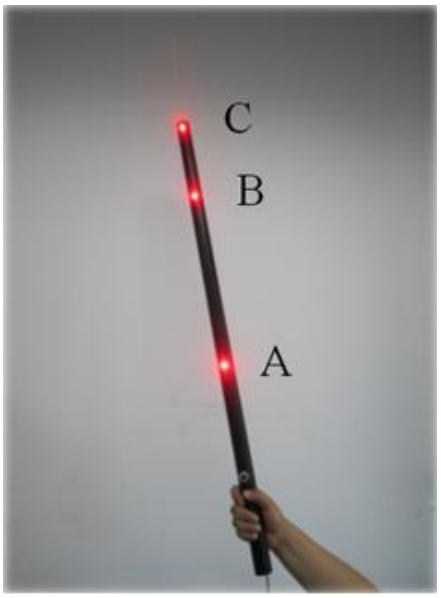
The 3D calibration objects are not as easily accessible compared to the planar objects mainly due to the significant increase in complexity and precision required for fabrication. With regard to practicality, 3D objects are also difficult to carry through an observation volume and also have a limited area of application due to the volume and size of the 3D object [20]. Nevertheless, 3D calibration objects can provide accurate camera calibration [20][24] and the expense of the approach can be justified if the level of accuracy requires it.

1-Dimensional Objects

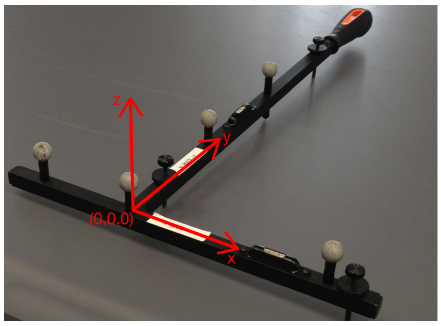
1D patterns or calibration wands are common names used to refer to any object that is defined along a single axis. A wand is most often made up of at least 2 collinear identifier points at defined locations. There are cases that deviate slightly from the category name "1-Dimensional", where a calibration object referred to as a wand takes the form of a T-shape, such as Vicon's Active Wand [11]. It is also common to find configurations of two or three orthogonal 1D rigid bars to be referred to as a calibration wand, an example of this configuration is shown in Figure 2.5a [23].

Calibration wands which are fabricated from a rigid bar and spherical markers can introduce a new problem. When a sphere is projected onto an image plane it is likely to be distorted in the sense that the resulting projection is not always a circle or an ellipse. This complicates the procedure of finding the centeroid of the spherical marker and makes it difficult to consistently identify the same point on the wand maker [25].

Zhang [21] was one of the initial individuals to propose a method for camera calibration using a 1D pattern and made specific mention on the suitability of the method in situations where cameras faced each other or did not have an overlapping field of view. There were limitations to Zhang's initial approach as the 1D pattern had to be fixed at one end in order to solve the system of equations. Notably though, this approach only made use



(a) Rigid 1D bar with LED lights at defined points [18]



(b) A T-shaped wand used in the Vicon system[11]

Figure 2.6: Figures showing the variations in calibration objects which are referred to as wands or 1D patterns

of one camera.

Calibration with a 1D patterns has since progressed and there are a variety of methods to do so. Mitchelson et al. [26][25] propose methods based off Hartley's [27] pairwise camera calibration and impose no restriction on the movement of the wand or any necessity for the camera field of views to overlap. Pribanic [23] also makes use of wands but with two and three orthogonal rigid bars and was found to produce results which are comparable and better in some cases than that of a commercial 3D kinematic system.

The simplicity and ease of mobility associated with 1D patterns makes then a desirable option for quick and efficient camera calibration [23][26]. Furthermore, the fabrication process does not seem as extensive in comparison to 3D objects but there is room for professionally machined wands, if the demanded level of accuracy requires it. The wand calibration method is limited with regards to the amount of information returned per image, where the limiting factor is the number of points on the wand. This does not mean a wand cannot be used to calibrate cameras, it means that the method is more susceptible to noise from images as there are only a few points per image [20][21].

2.3 Sparse Bundle Adjustment

The sparse bundle adjustment (SBA) is an optimization method utilized commonly to refine a camera calibration [17][18]. The SBA is typically based off a nonlinear least-squares solver intended to minimize the reprojection error. A least-squares method that is often referred to in the literature is the Levenberg-Marquardt algorithm [17][25].

A least-squares algorithm attempts to find optimal parameters for a mathematical model given a set of data points. The function operates through trying to minimize an objective which is commonly the sum of the squares of errors between the data points and the output model data points; model data points are a resultant from a certain set of parameters input to the model. The algorithm is computed in an iterative fashion, adjusting the parameters of model in an attempt to find the minimum objective.

A typical objective could be expressed in the following way

$$\min_{\hat{P}} = \sum_{i}^{m} C(e_i) = \sum (m_i - \hat{m})^2 \tag{2.20}$$

So in reference back to the SBA, it attempts to optimize the 3D point positions which have been calculated from the observed pixel data. The parameters associated with the points are adjusted simultaneously with the intrinsic and extrinsic parameters of the camera in an attempt to minimize the reprojection error. The reprojection error refers to the residual or difference in pixels between the 3D estimated points which have been reprojected onto the image plane and the input pixel coordinates of the observed points.

2.4 Convolution Neural Networks for Feature Detection

Machine learning has become a term that most individuals have heard of in our world today simply because the possible applications are becoming evermore realisable due to technological development and improvement of high performance computing. In particular, a subsection of machine learning that is referred to as Convolution Neural Networks (CNNs) have seen increased popularity and performance for applications associated with computer-based pattern recognition from videos and images [28].

To explain simply what a CNN is, it can be thought of as a number of consecutive

refinement processes which have particular "filters" to identify certain characteristics from an image or other form of data. There are a number of different kinds of refinement filters in a CNN and they are commonly arranged in an order of simple ones to more complex ones. Some examples of simple filters are edge detectors that will either be specifically designed to find left, right, top or bottom edges of a pattern in data. Thus once a network has been developed, data can be passed to it and the filters will identify if the desired information is amongst the entered data.

An unfortunate characteristic of neural networks and deep learning is they often require significantly large data sets in order to generate neural networks which can consistently identify objects in the variable real world [28]. A technique known as *transfer learning* is often proposed as a good approach to overcome the need for excessively large data sets. The reason being is, a CNN that was trained on a large data set to perform a specific task can effectively be "re-purposed" and perform a new task that is trained on a smaller data set. This is a cost effective way to save time and means that deep learning techniques can be used on smaller studies which do not have access to extremely large data sets [3].

The reason that this technique is applicable, is during the training process the neural network learns mechanisms to identify features in the observed data. The features of the data can often be broken down into a composition of filters which allow the neural network to identify the features. These learnt features are sometimes transferable between similar studies and there is therefore no reason to retrain a CNN to re-learn those similar features [4].

Thanks to the increasing amount of research into CNNs and the benefits of re-purposing networks, it has created room for a number of new study opportunities particularly around the motion of people and animals. Specialist deep learning networks such as DeeperCut have been developed to analyse the pose and motion of humans [3]. It is important to note that in the context of studying people and animal motion, *pose* describes a particular position that the subject takes on in the viewed frame, it does not have the same meaning as a camera pose described in previous sections. Nath et al. [3] has previously shown that trained networks can be re-purposed to work on studies based on smaller data sets. They particularly did this with DeeperCut and have since developed their own pose estimator DeepLabCut, which is particularly accommodating for studies with limited sized data sets.

Figure 2.7 shows some resulting outputs from training DeepLabCut to locate and identify certain points on observed subjects. The coloured markers are all unique and identify specific positions on the subject, these makers will always highlight the same point unless of course the point is obstructed.

Figure 2.7: DeepLabCut analysing the pose of distinctly different subjects, the coloured markers are user defined and track specific motion or behaviours [3].

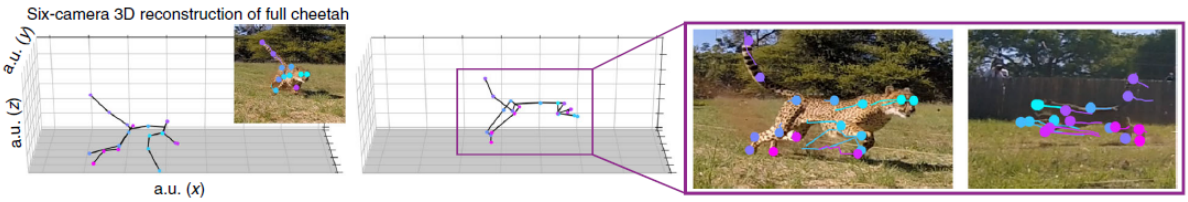


Figure 2.8: A 3D reconstruction of the data points extracted by DeepLabCut from a 6 camera view to further study the motion of cheetah [3]

In the case of multi-camera motion capture, once the points have been retrieved from the trained DeepLabCut network, it is then possible to reconstruct the motion and behaviour, similar to that of Figure. 2.8.

The typical process for training a CNN like DeepLabCut can be broken down into distinct steps.

1. Data collection
   (a) It is advised to capture data with as much variability as possible [3].
   (b) This includes observing as many of the possible poses in the motion range and doing so under different environmental conditions with varying backgrounds or lighting.
2. Label data consistently
3. Train the CNN on the labeled data
4. Evaluate the CNN

(a) After training, the CNN's performance will be evaluated to determine if the CNN produces the desired accuracy.

(b) If the accuracy does not meet the required standard then, wrongly predicted data will be corrected and passed back as additional data to retrain the CNN [3]

This encompasses the iterative process of training a CNN.

2.5 Trajectory Based Optimization

This section is intended to give a brief over view of what trajectory optimization is, the theory required to formulated and understand a problem and lastly, make reference to some applications as described in the literature.

2.5.1 Defining a Trajectory Optimization Problem

In order to understand what trajectory optimization is, one first needs to understand what a *trajectory* is. A trajectory is commonly a path of motion between a start and ending point, where the path of the trajectory is defined by the time dependent states, $\mathbf{x}$, (position and velocity of the dynamic model) given some control variables, $\mathbf{u}$, (applied force) [29].

Trajectory optimization, attempts to find the optimal path between the start and end point by manipulating the input control parameters of the dynamic model while simultaneously trying to minimizing a cost function.

An optimization problem is commonly composed of the following aspects [30]:

$$\text{Optimal Trajectory:} \quad \{\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\} \tag{2.21}$$

$$\text{System Dynamics:} \quad \dot{\boldsymbol{x}} = f(t, \boldsymbol{x}, \boldsymbol{u}) \tag{2.22}$$

$$\text{Constraints:} \quad \boldsymbol{c}_{min} < \boldsymbol{c}(t, \boldsymbol{x}, \boldsymbol{u}) < \boldsymbol{c}_{max} \tag{2.23}$$

$$\text{Boundary Conditions:} \quad \boldsymbol{b}_{min} < \boldsymbol{b}(t_0, \boldsymbol{x}_0, t_f, \boldsymbol{x}_f) < \boldsymbol{b}_{max} \tag{2.24}$$

$$\text{Objective Function:} \quad J = \phi(t_0, \boldsymbol{x}_0, t_f, \boldsymbol{x}_f) + \int_{t_o}^{t_f} g(t, \boldsymbol{x}, \boldsymbol{u}) dt \tag{2.25}$$

2.5.2 System Dynamics

The dynamics of the system can generally be broken down into a set of linear or non-linear equations which describe how the system changes with time. This is one of the fundamental constraints in optimization as it tailors the problem to obey the system dynamics and produce associated results [29].

In Clark's [4] work on 3D markless motion capture, he models the dynamics of a cheetah to obey a constant acceleration model,

$$\ddot{x}_k = \ddot{x}_{k-1}$$

where $\ddot{\mathbf{x}}_k$ represents an acceleration at time instance, $\mathbf{k}$. However, a cheetah is complex multi-link moving quadruped and the observed motion would not realistically obey a constant acceleration. So in order to overcome this, an error term, $\mathbf{w}_k$, was included and termed the acceleration error. This meant that a dynamic model constraint could be passed to the optimization problem to limit the way the system changes over time.

Coordinates and Reference Frames

When defining the dynamics of a system it is important to identify the two reference frames, one being the inertial or world reference frame and the other, the body reference frame. The body reference frame is fixed to the single or multi-bodied system and is used for convenient reference of the different members to each other. Diebel [31] provides an excellent explanation of the considerations required to define the pose of a rigid body, some necessary terms will be covered here.

In order to relate the inertial and body reference frame to one-another, either reference frame needs to undergo a rotation, $\mathbf{R}$, and the point of the reference frame origin needs to be known.

Given a vector, $\mathbf{v}$, a rotation matrix, $\mathbf{R}$, will rotate the vector by a given angle about a certain axis and preserve the length of the vector. A Euler-angle rotation about the z-axis through an angle, ψ, is shown as an example.

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.26}$$

Euler-angle rotations provide an easy means to define the orientation of a rigid body through three consecutive rotations [31]. For instance, transitioning from the inertial frame, represented by **I**, to the body frame, represented by **0**, could be done by first rotating about the x-axis, then the y-axis and finally the z-axis to achieve the final orientation.

$$\mathbf{R}_I^0 = \mathbf{R}_z(\psi) \times \mathbf{R}_y(\theta) \times \mathbf{R}_x(\phi) \tag{2.27}$$

Some notable properties of rotation matrices make transitioning between two reference frames relatively easy [31].

$$determinant(\mathbf{R}) = 1 \tag{2.28}$$

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.29}$$

Equations (2.30) and (2.31) can be used to transition a point, **P**, in the inertial frame to the body frame and vice-verse.

$$\mathbf{P}_0 = [\mathbf{R}_I^0]\mathbf{P}_I \tag{2.30}$$

$$\mathbf{P}_I = [\mathbf{R}_I^0]^T\mathbf{P}_0 = [\mathbf{R}_0^I]\mathbf{P}_0 \tag{2.31}$$

2.5.3 Constraints

In an optimization problem it is important to define limitations for the system and how it can operate. Clark [4] explains that imposing constraints on a system is one of the benefits that trajectory optimisation has over other state estimation techniques, especially in the area of bio-mechanics and limiting motion of joints to their normal operating range.

As described above, the dynamics of a systems is considered to be one of the constraints of a trajectory [29].

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \tag{2.32}$$

A trajectory optimization can also include path constraints to limit the trajectory of the problem.

$$\mathbf{h}(t, \mathbf{x}(t), \mathbf{u}(t)) \leq 0 \tag{2.33}$$

Non-linear boundary constraints and constant limits are further types of constraint which can be imposed if applicable. Boundary constraints according to Kelly [29] intend to limit the initial and final states of a system. Whereas, constant limits are used to define an operating range for states and variables, such as a rotation range or maximum input applied force.

$$\text{Boundary Conditions:} \quad \mathbf{b}(t_i, t_f, \mathbf{x}_i(t_i), \mathbf{x}_f(t_f)) \leq 0 \tag{2.34}$$

$$\text{Constant Limits:} \quad \theta_{min} \leq \theta(t) \leq \theta_{max} \tag{2.35}$$

$$0 \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{2.36}$$

Further explanation on the types of constraints and others which are not applicable in this case can be found at [29].

2.5.4 Objective Functions

Kelly [29] describes an objective function to be a mathematical way to represent what is meant by the "best" feasible trajectory and how to classify it. Before progressing, an important term must be defined. *Decision Variables*, are parameters that an optimizer can adjust to try and minimize the objective function.

Typical objective functions associated with modeling dynamics systems are termed the minimum force squared and the minimal absolute work functions.

A general objective function has two terms, a boundary objective, $J(.)$, and an integral path along the entirety of the trajectory, $w(.)$ [29]. There are three forms of objective function, one which contains both terms, $J(.)$ and $w(.)$, and two other forms which have either the boundary or integral path objective.

$$\min_{t_i, t_f, \mathbf{x}(t), \mathbf{u}(t)} J(t_i, t_f, \mathbf{x}_i(t_i), \mathbf{x}_f(t_f)) + \int_{t_i}^{t_f} w(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau)), d\tau \tag{2.37}$$

In the work of camera calibration, the typical objective function is often based on minimizing the residual error between measured image points, $\mathbf{x}_j^i$, and 3D estimate points, $\hat{\mathbf{X}}_j^i$, that have been reprojected to image points, $\hat{\mathbf{x}}_j^i$, [23].

$$\min_{\mathbf{P}^i, \mathbf{X}^i} \sum_{i,\, j} d(\mathbf{x}_j^i, \hat{\mathbf{x}}_j^i)^2 = \sum_{i,\, j} d(\mathbf{x}_j^i, \mathbf{P}^i \hat{\mathbf{X}}_j^i)^2 \tag{2.38}$$

Where, $\mathbf{P}^i$, is the projective matrix for the i^{th} camera. The camera calibration literature also makes reference to some objective functions that include the intrinsic parameters, $\mathbf{K}$ and $\mathbf{D}$, as part of the decision variables that the optimization can manipulate [4].

Cost Functions

A common problem within optimization is the effect of outliers skewing the resultant solution [4]. An outlier, in the sense of camera calibration, could be defined as a large reprojective error resulting from a poor estimation. These large outliers have a cost associated with them and have varying levels of effect depending on the cost function used in the optimization problem.

Mactavish [6] comments on the effect of outliers in robotics navigation and suggests a solution known as Random Sample Consensus (RANSAC). RANSAC, operates by solving a model with the minimum required number of randomly selected data points. It then, tallies the number of random model iterations that "agree" with each other and finally the largest data set that agrees with each other is used in the process. However the approach does have some pitfalls as it is sensitive to "inlier" thresholds. Meaning if the inlier threshold is set too close the ideal values, then valid data points which deviate slightly will be rejected but set the threshold to high and outliers will be included.

Another approach to deal with outliers as suggested by Mactavish et al. [6][4], is through the use of robust cost functions. A robust cost function is one which limits the cost of outliers but can preserve the influence of valid measurements. For instance, the cost function shown in equation (2.38), has a quadratic relationship associated with it and therefore the larger the error the higher the cost. This can be altered by using a different cost function with a different relationship and thereby reducing the effects of large errors [4].

A notable cost function is known as the redescending cost function with three regions defined in a piece-wise manner and thus varying cost dependent on the magnitude of the error [6][4]. The region of particular note is where a constant cost is defined, thus when a threshold error is reached the cost will no longer increase. This cost function is used to seemingly nullify the effect of outliers [4].

Varying cost functions each have their own benefits and drawbacks [4] and should be selected carefully for the particular problem at hand.

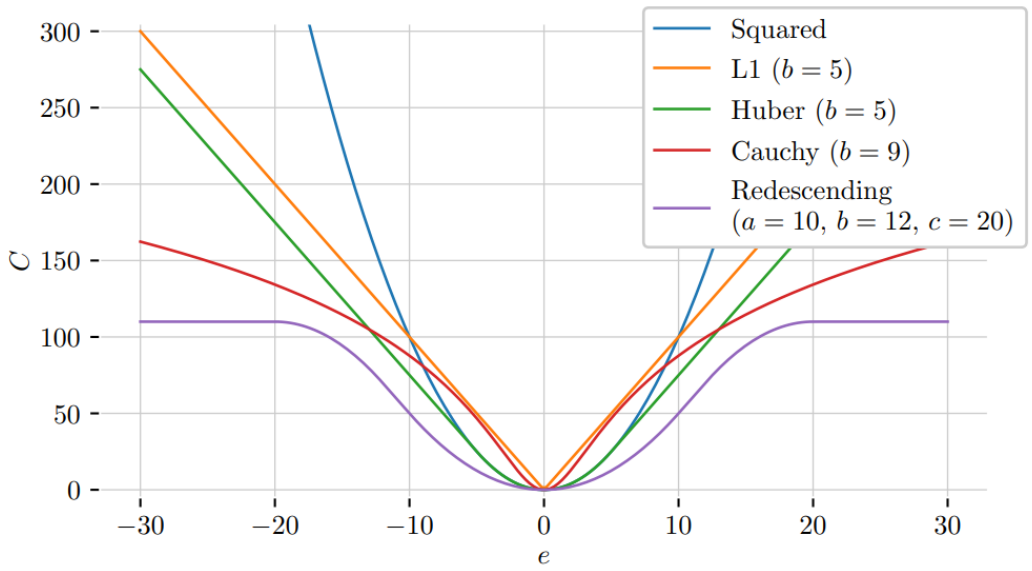


Figure 2.9: A comparison of five common cost functions, highlighting the differing influences of large errors. It is clear that the redescending function limits the effect of large outliers by maintaining a constant maximum cost [4]

Chapter 3

Methodology

This chapter is intended to outline the development of particular task to achieve camera calibration and the information

3.1 Project Development Strategy

Having reviewed the information available with regards to camera calibration, a development strategy could then be formed to clearly define the problem statement and identify requirements to meet it.

3.1.1 Problem Statement

This research project aims to identify a suitable technique for automatic multi-camera extrinsic calibration, specifically in scenarios of video capture outdoors, with camera arrangements which surround a subject and may not have over lapping fields of view (FOV). We intend to thus define and fabricate a suitable calibration object which can provide sufficient data for all surrounding angles in a large outdoor volume. This should be coupled with some form of identification method which is capable of tracking the calibration object and extracting image data. Lastly the extracted data should be used to determine optimal camera extrinsic parameters.

3.1.2 Project Tasks and Requirements

Given a clearly defined problem statement, the project was then broken down into three main tasks. Within each of these tasks, specific requirements were created to guide the decision making and ensure development was kept within a desired scope.

The major tasks associated with this project are defined below and mostly occurred in the order that they appear.

Appropriate Calibration Object:
This related to evaluating the various applicable calibration patterns according to the feasibility of fabrication, accuracy performance and suitability for camera arrangements

which surround a subject.
The desired requirement of the calibration object were set out:

1. Mobility, the object should be easy to manipulate and clearly visible in a volume of approximately $12m \times 6m \times 3m$.
2. The object must be visible from all angles.
3. Sufficient data should be acquired to perform extrinsic calibration.

Locating the Calibration Object:
Dependent on the calibration object chosen, a suitable approach of identifying and extracting information about the object from image data had to be explored. This involved identifying possible object recognition methods that already existed, be it either complete software or training a convolution neural network to perform a similar duty. This task had significant dependencies on the calibration object chosen.
Thus the requirements were:

1. Robust tracking and identification of the calibration object.
2. Return 2D pixel data associated with the tracked points on the calibration object.
3. The system should automatically identify the calibration object.

Camera Calibration:
Camera calibration referred to determining the intrinsic and extrinsic parameters of a camera arrangement and the possible algorithms to do so. It also included identifying a technique which could be used as a ground truth for comparison and evaluation. Open source and proprietary software options where investigated to understand the suitability of the chosen calibration object and ground truth object.
The requirements of calibration were:

1. Employ a suitable ground truth calibration technique for comparison.
2. The calibration is mainly intended to determine the extrinsic parameters.
3. Calibration parameters should be optimized to return an accurate calibration that is comparable to the ground truth test.

3.2 Design Overview

Having identified a clear problem statement with relevant requirements, some fundamental design decisions could be made that would influence the remainder of the project. This will highlight the important design decisions and give some reasoning behind them. Design development and iteration will be covered in a later section.

Calibration Wand:
The calibration object chosen took the form of a 1D pattern, for ease of reading, this will be referred to as a wand going forward. The wand was composed of a single axis with three identification spheres located at the top, middle and bottom of the rigid rod. It was decided that the structure of a wand provided the most suitable combination of characteristics to meet the requirements defined above.

Note that a 1D pattern was preferred over a 3D pattern due to the increased feasibility in fabrication of a wand compared to other 3D structures. Fabrication of precise 3D structures would require a large amount of time, and the lockdown restrictions of COVID-19 was deemed a factor that would exacerbate this issue further. In the end, the goal of the project was not to create the "best" calibration object but instead, attempt to achieve a suitable level extrinsic calibration accuracy.

Some additional motivating factors to make use of a wand:

- It followed a simple design that could be easily revisited with multiple iterations if necessary.
- Practically, a wand shape is far more easy to store, transport and maneuver in comparison to other calibration techniques, especially when the calibration object is required to be distinctly visible in a volume of $12m \times 6m \times 3m$.

Furthermore, it was decided that the wand calibration would be compared to a ground truth system based on 2D pattern calibration, the ground truth was based off Zhang's algorithm [10].

Wand Identification:
After some investigation, it was decided that a CNN would be a suitable approach to automatically track and identify the calibration wand. Specifically, DeepLabCut [3] was used to track the wand and return pixel data associated with the three identification markers. The fact that DeepLabCut could be used to train a CNN on a relatively small data set in comparison to other deep learning studies, was particularly favourable. DeepLabCut

was further motivated for, as it was easy to create multiple iterations, correcting and adding data to improve the robustness of the CNN as well as allowing us to improve the accuracy to a desirable level. Other options which were considered was EasyWand [32], however, this system was limited in the sense that it only used calibration wands with two identification points and did not offer actual feature tracking.

Calibration Method:
Typically camera calibration undergoes two processes, this includes determining initial estimates for the intrinsic and extrinsic parameters and then further optimizing those parameters through a sparse bundle adjustment (SBA), [23][26]. A major focal point of this project was to investigate the possibilities of performing an extrinsic calibration through the method of trajectory optimization. Trajectory optimization was utilized as it incorporated temporal information into the optimization of the calibration parameters. This meant that consecutive frames in a video were related to each other via the dynamics of the calibration wand, limiting the movement to feasible paths.

3.3 Cameras

This section identifies specific information regarding the cameras used throughout the various processes involved in the project. It notes the typical arrangements of the cameras used in data capture for training the neural network and for the purpose of extrinsic calibration.

3.3.1 Specifications

Throughout the project, two types of cameras were used to capture video footage. The video footage was used in either training DeepLabCut or to capture point information for the camera calibration processes.

The two types of camera used were the GoPro HERO Session 5s (GoPro HS5) and the mobile camera on an Iphone 6S. During filming, the GoPro HS5 was set to shoot in wide angle mode to increase the FOV, the specific setttings can be seen in Table 3.1

Table 3.1: Specific camera settings used throughout the project for the GoPro HS5 and Iphone 6S.

Camera Type	Resolution	Frame Rate (fps)	Horizontal FOV (°)	Vertical FOV (°)
GoPro HS5	1080p	90	118.9	69.7
Iphone 6S	1080p	30	58.632	

Initially, training data for DeepLabCut was captured using two Iphone 6S cameras, the main reason being an accessibility issue. However after a point, video capture was moved to the GoPro HS5s as they were to be used to capture the calibration data.

3.3.2 Video Footage

Over the course of the project, video footage had to be recorded for two main reasons. The first was to capture data of the calibration wand over a variety of movements, this was used to train DeepLabCut. Secondly, multi-camera footage had to be taken of the calibration wand and checkerboard 2D pattern to determine extrinsic parameters of the cameras. The subsequent part of this section defines the camera arrangements used and the need for footage synchronization.

Camera Arrangements

In general, the camera arrangements were set-up in a relatively arbitrary fashion when video footage was being recorded for training DeepLabCut. The reason being, that each set of video frames was evaluated individually and no information had to correspond between the videos. Furthermore, arbitrary arrangements were employed to introduce some variability into the training data as this increased the variety of orientations that the calibration wand was viewed from.

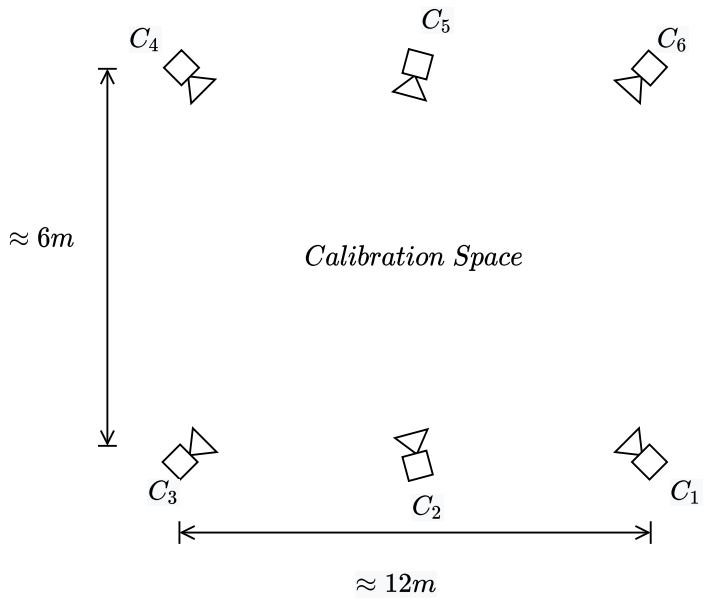


Figure 3.1: Cameras arranged to surround an observed volume for multi-camera footage capture.

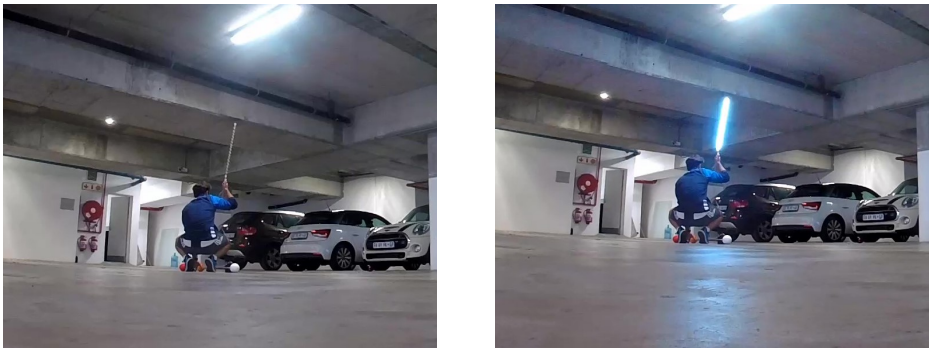
With regards to multi-camera video footage for extrinsic calibration, the cameras were

arranged in more structured manner. The arrangement was chosen to surround a "calibration space" that would effectively be seen from all sides. Notably, as in Figure 3.1, cameras C_2 and C_5 do not face directly towards each other. This was done as a precaution to avoid possible depth of view problems where C_2 and C_5 provide conflicting information about where a point is in 3D space.

Camera Synchronization

As explained the, video footage for DeepLabCut training did not need to be synchronized as the footage from each recording was treated as separate input. However, in the case of mutli-camera calibration, there was indeed a need to synchronize the cameras.

From the camera arrangement in Figure 3.1, it is evident that footage synchronization is not possible when a single person is starting the recording of each camera. Synchronization was thus facilitated by flashing a light at the start and end of each video. A light rod shown in Figure. 3.2 was used to synchronize all the cameras as it was visible from all directions. This allowed the recordings to be clipped to the same number of frames in a post recording stage.



(a) Footage frame before light rod on.

(b) Subsequent frame with light on

Figure 3.2: Two consecutive frames from video footage showing the use of the light rod to synchronize multiple cameras.

The light rod was composed of simple materials, where the crucial components were: a 14.4W/m LED light strip, a switch and a 12V 7Ah lead acid battery to power it. A notable factor was the high power of the LEDs, 14.4W/m, in order to be visible in a bright outdoor scene.

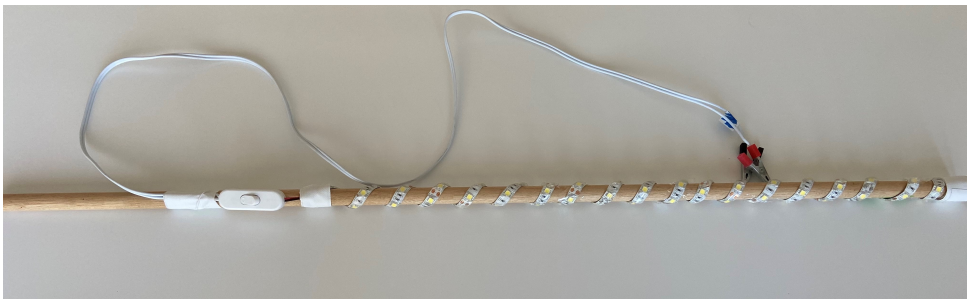


Figure 3.3: Light rod used to synchronize video footage

3.4 Checkerboard Calibration

The checkerboard calibration served as a ground truth to evaluate the extrinsic calibration achieved in the trajectory optimization. The following section outlines the process used to determine the intrinsic and extrinsic parameters for the GoPros in an arrangement shown in Figure 3.1.

3.4.1 Intrinsic Calibration

The intrinsic parameters are influenced by the internal design of a camera and fortunately remain mostly constant unless, the focal zoom is altered.

The first step to calibrate the intrinsic parameters of a camera is to extract a number of frames from a video. One hundred frames of the checkerboard were extracted from a single video using a script written in Matlab. During the process of extraction, frames were selected such that the checkerboard covered a large area of the FOV in varying orientations and positions.

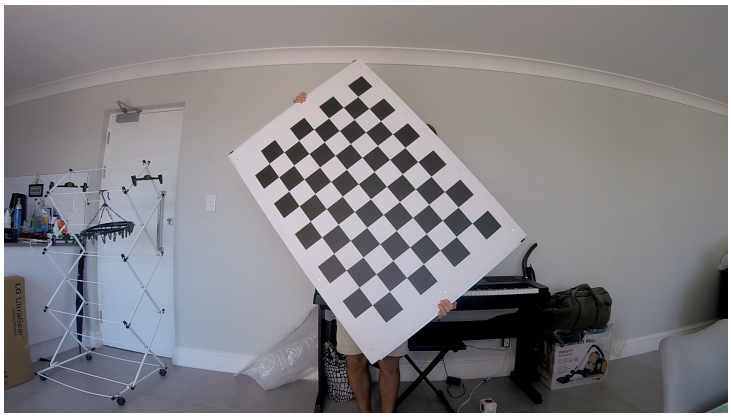


Figure 3.4: A frame captured by the GoPro HERO Session 5 that was used in the intrinsic calibration process. The distortion of the GoPro fisheye lens is apparent as straight lines seem to curve.

The extracted frames were then input into Matlab's Camera Calibration toolbox [22]

to identify and locate the corners within the checkerboard pattern. A 3D plot of the checkerboard positions from Matlabs camera calibration can be seen in Figure 3.5. The checkerboard has two patterns within, one being the total number of squares and the second being the total number of corners. A corner is defined as point where two black squares meet and two white squares meet. The checkerboard used in this project had 11×8 squares and 10×7 corners, with a square length of 90mm.

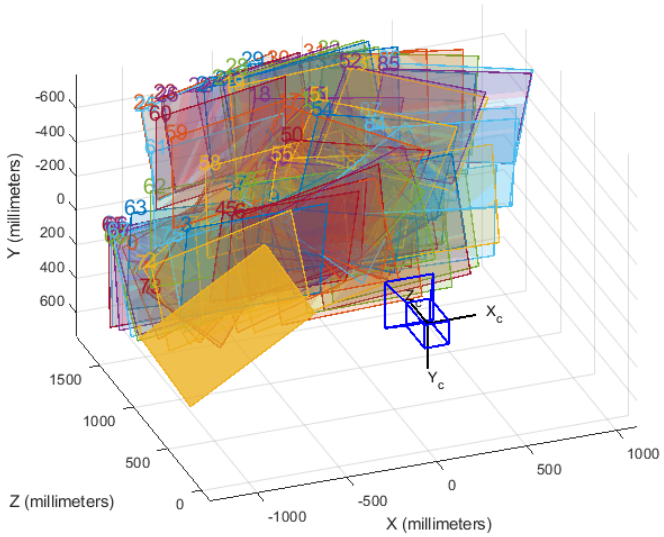


Figure 3.5: A Matlab plot showing the various 3D orientations of the checkerboard during intrinsic calibration with respect to the camera

The Matlab checkerboard points were then converted to a suitable file to use in OpenCV's camera calibration library [1]. Since the GoPro had such a large FOV and significant radial distortion the specialized fisheye calibration functions were used to cater for the distortion. Figures 3.6 clearly shows the significant distortion associated with a fisheye lens, this can be seen as the rectangles seem to have rounded edges instead of straight edges.

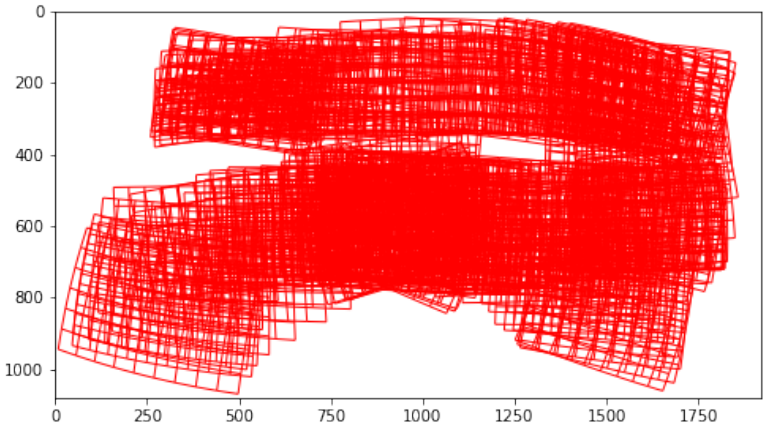


Figure 3.6: Distorted overlaid frames from Matlab

The distorted points and known dimensions of the checkerboard were entered into OpenCV's fisheye calibration function to determine the camera calibration matrix, **K**, and distortion parameters, **D**.

The calibration function returns the following intrinsic parameters:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & pp_x \\ 0 & f_y & pp_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 885.21 & 0 & 963.14.72 \\ 0 & 893.17 & 551.84 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\mathbf{D} = [k_1, k_2, k_3, k_4] = [0.11533794, -0.03716893, 0.00106535, 0.0007212] \tag{3.2}$$

The resulting root mean square (RMS) reprojection error:

	Value
RMS Error	0.281 Pixels

After determining the **D** parameters, the extracted checkerboard points could be undistored to further show the extent of radial distortion in fisheye lenses.

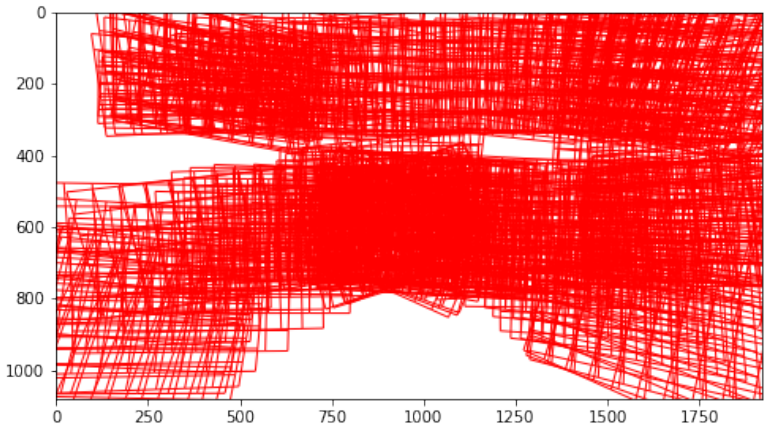


Figure 3.7: OpenCV undistored overlaid frames

3.4.2 Extrinsic Calibration

In order to facilitate extrinsic calibration with a checkerboard and the camera arrangement in Figure 3.1, the cameras had to calibrated according to a pairwise method. This meant that the checkerboard had to be moved in front of two consecutive camera pairs, such that a set of points were seen simultaneously by the camera pair. For example, cameras C_1 and C_2, must have seen the checkerboard in their overlapping field of view to

develop a set of corresponding points. This process was repeated for C_2 and C_3, C_3 and C_4, etc.

At the start of recording, the LED light rod shown in Figure 3.2 was flashed and the 11×8 checkerboard was subsequently moved around the calibration space. The checkerboard was manipulated to cover as much of the volume in as many orientations as possible to develop a good calibration. The LED light was then flashed again at the end of the calibration movement.

The frame extraction followed the same procedure as mentioned in the intrinsic calibration. The frames were extracted by a Matlab script and the corners of the checkerboard were detected using the Matlab Camera Calibration toolbox [22].

The extrinsic parameters of the camera were calculated using OpenCV's *stereoCalibrate* function. This function computes the camera pose ($\mathbf{R}$ and $\mathbf{t}$) between consecutive camera pairs given a set of 2D corresponding points. Thus given the pairwise manner of creating point correspondence between successive camera pairs, the relative poses between each camera could be determined. Once all the relative camera poses are known and the pose of one camera is known, then each camera can be related to any other camera.

To explain further, assume the pose in world frame of C_1 is known. If C_2 is at a known pose relative to C_1 and C_3 is at a known pose relative to C_2 then, C_3 can be related to C_1.

Once the relative camera poses were known they could all be related to the world reference frame. In this case, the world reference frame coincided with camera one, C_1.

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.3}$$

$$\mathbf{t}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.4}$$

The camera poses were related to each other in the following way:

$$\mathbf{R}_{i+1} = \mathbf{R}_{i,i+1}\mathbf{R}_i \tag{3.5}$$

$$\mathbf{t}_{i+1} = \mathbf{R}_{i,i+1}\mathbf{t}_i + \mathbf{t}_{i,i+1} \tag{3.6}$$

Where $\mathbf{R}_i$ and $\mathbf{t}_i$ denote the camera pose in the world frame and the subscript, $(i, i+1)$, refers to the relative camera pose.

Relating all the camera poses to the world reference frame essentially concluded the extrinsic calibration process and served as an initial estimate. Before performing the SBA, the extrinsic parameters and checkerboard points were plotted to serve as a sanity check. The resulting reconstructions can be seen below in Figures 3.8 and 3.9.

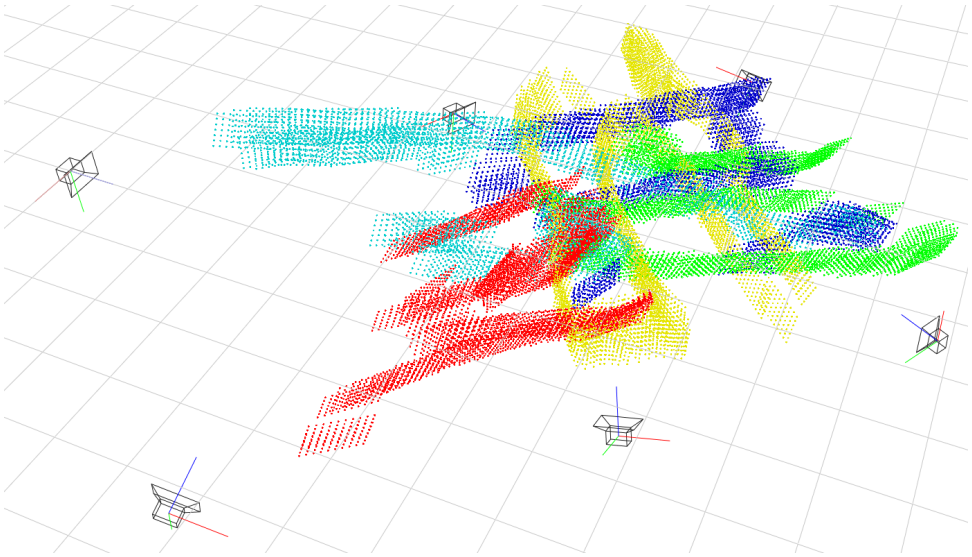


Figure 3.8: Checkerboard 3D point reconstruction following the initial estimate of the extrinsic parameters. Point correspondence between cameras (Camera 1 is in the top left): Red(1&2), Green(2&3), Yellow(3&4), Blue(4&5), Cyan(5&6).

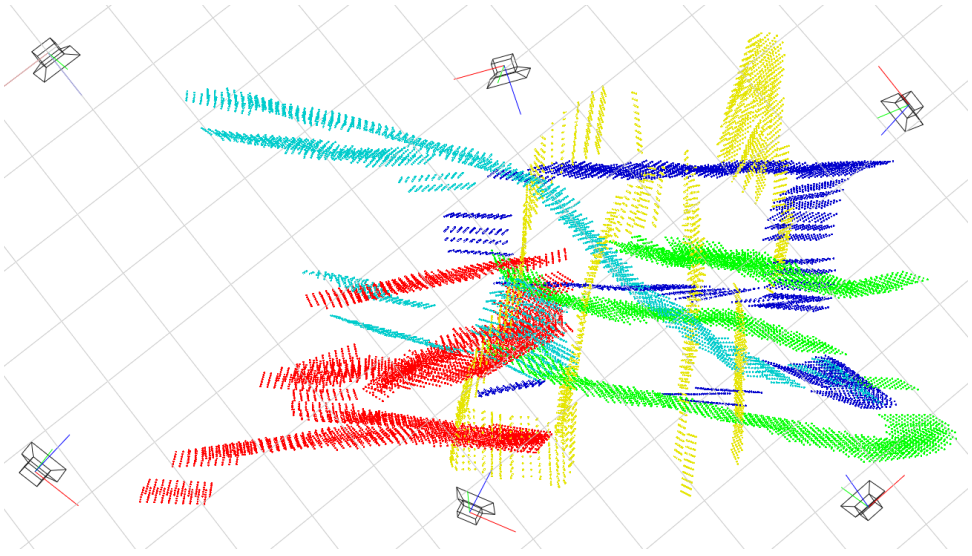


Figure 3.9: Birds-eye view of the checkerboard 3D point reconstruction following the initial estimate of the extrinsic parameters.

3.4.3 Sparse Bundle Adjustment

As a final step of the extrinsic calibration, a sparse bundle adjustment is often performed to optimize the parameters by minimizing the reprojection error [4][10]. Before performing the SBA, the initial estimates of the intrinsic and extrinsic parameters were used to

generate 3D estimate points from an OpenCV triangulation function. These 3D estimate points, $\hat{X}_j$, were then reprojected to pixel points, thus termed *reprojected 2D pixel points*.

The sparse bundle adjustment was performed by firstly defining a residual error. The residual was the difference of the measured 2D pixel points, $\mathbf{x}_j^i$,and the reprojected 2D pixel points, $\hat{\mathbf{x}}_j^i$.

$$e_j = x_j - \hat{x_j} \tag{3.7}$$

The sparse bundle adjustment made use of a square loss function as seen in equation (3.8).

$$\min_{\mathbf{X}_i,\mathbf{R},\mathbf{t}} \sum_j C(e_j) = \sum_j (e_j)^2 \tag{3.8}$$

The residual error, decision variables, $(\hat{\mathbf{X}}, \mathbf{R}, \mathbf{t})$, and loss function were passed to a SciPy least-squares function to minimize.

A further note, the SBA was setup to minimize the reprojection error for all the cameras instead of just between a pair of cameras. This was done in much the same way as Clark [4] performed.

	Mean	Standard Deviation
Before SBA	-0.01167	0.26415
After SBA	-0.01547	0.33359

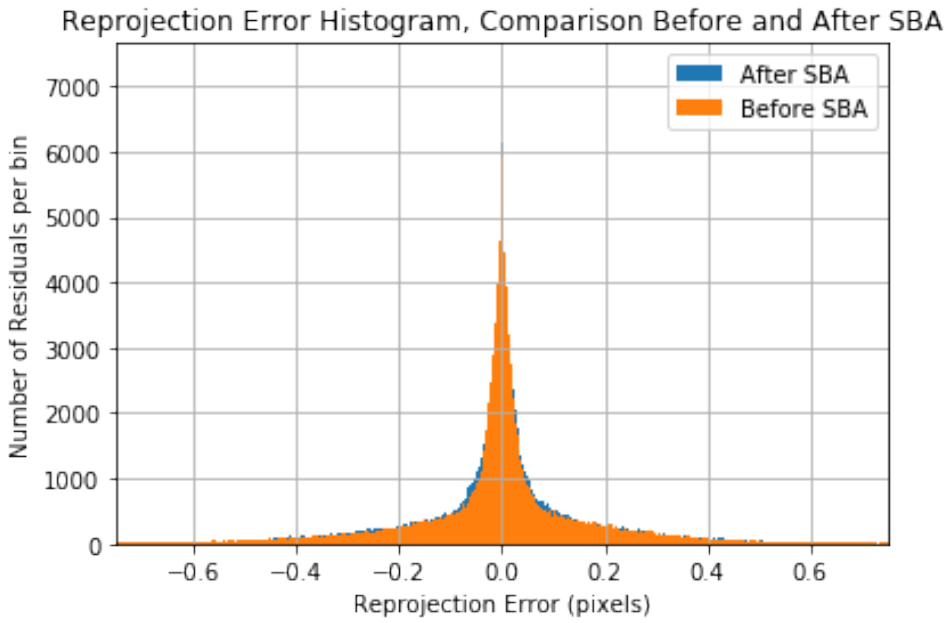


Figure 3.10: Histogram comparing the reprojection error before and after the sparse bundle adjustment.

Chapter 4

Model Design and Development

This chapter details the design steps and iterations taken for creating the calibration wand, training DeepLabCut and developing the trajectory optimization problem. These section were very dependent on each other and essentially followed a linear cycle.If revisions were required, then sections which followed were also required to be revisited or adjusted.

4.1 Wand Fabrication

It was decided that the calibration object should take the from of a 1D pattern. From here forward the calibration object will be referred to as a wand. From reviewing the work of Michelson et al. [26], it was determined that a 1D calibration object could be composed of two identification markers along a straight rigid rod. However, it was decided to introduce a third identification marker simply because the more points that used in a calibration process, the less susceptible the method is to noise [21].

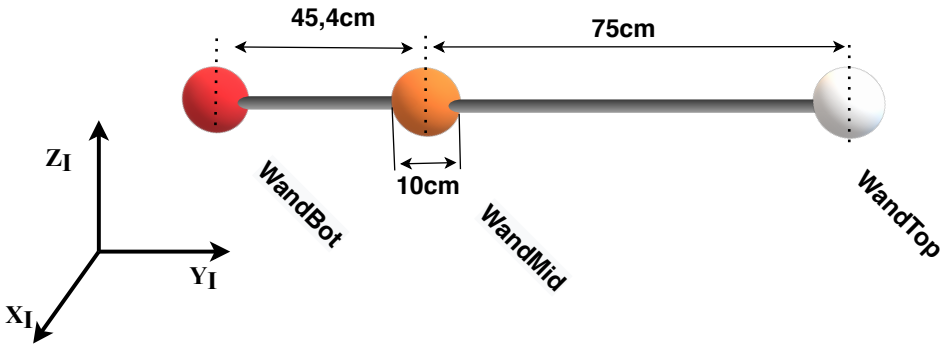


Figure 4.1: Three dimensional illustration of the final calibration wand with the 3 identification markers.

The calibration wand was composed of simple materials. The identification markers were chosen to be 100mm in diameter so that they could be seen at distance approximately 12m away. The size of 100mm was inferred from work performed by Clark [4], who utilized 88mm checkerboard grids at distance around 10m away from a camera. The centroids of the top and bottom markers were taken to be at same point as each end of the rigid rod.

There were two iterations of the calibration wand. The final design can be seen in Figure 4.1. In the initial iteration of the calibration wand all three of the identification makers were white. Notably, the middle marker was placed with a bias towards one end. The

Table 4.1: Bill of Materials for the Calibration Wand

Quantity	Item
3	100mm polystyrene ball
1	$16mm \times 1200mm$ dowel stick
2	Acrylic paint

thought behind this had particular application when all three markers were white as it would allow consistent labeling of the markers, i.e the bottom marker was always the marker closest to the middle marker.

Upon reviewing the performance of the neural network which was trained to identify the first iteration of the wand, it was made clear that further differentiation was required between the markers. Thus colour was added to the markers to create a distinct difference between them. The colours of red and orange were deemed suitable as they are bright and would stand out in most scenes in the wild, allowing easy identification.

The final fabricated wand can be seen in Figure 4.2 below.

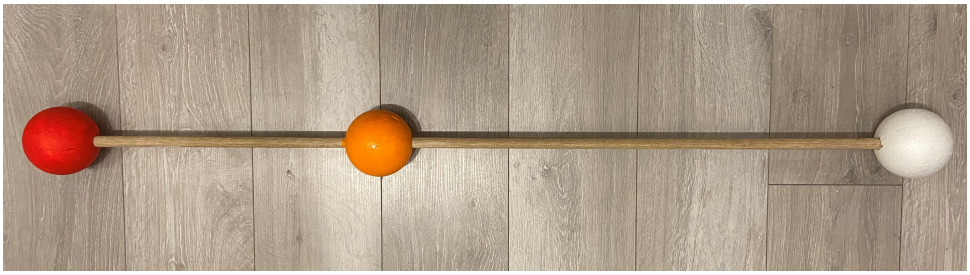


Figure 4.2: The final iteration of the calibration wand used to train DeepLabCut and in the Trajectory Optimization

4.2 DeepLabCut

DeepLabCut was trained to track and identify the calibration wand moving through an observed volume. In order to do so video footage was capture from two types of camera, an Iphone 6S and GoPros HS5s.

4.2.1 Setup and Process

Before any training could commence, the data needed to be labeled. All the data used in the project was labeled by hand using DeepLabCut's interactive GUI. Three labels were defined for the points on the wand; WandTop, WandMid and WandBot.

The initial training of the CNNs was performed on Iphone 6S video footage. There after, video footage record on the GoPros was analyzed using the CNN. If there were errors in the analyzed videos, the incorrect frames would be extracted, corrected and added to the initial set of frames to augment the data set. After data augmentation, the neural network would be retrained to incorporate the addition learning data.

Table 4.2: DeepLabCut CNN iterations over the entire project with the associated data set size, whiteWand refers to the first iteration of the wand and colourWand refers to the final version of the wand.

Training Iteration	Frames used	Videos	Frames/Video
whiteWand	480	24	20
colourWand 1.0	570	19	30
colourWand 2.0	990	33	30

During the training process, the data set being used was split into two sections. 95% of the labeled data would be used to train DeepLabCut and develop the required feature detectors. The remaining 5% of the data set would be completely unseen during training and used later to test the network and its performance, this was called the test data set. The training for all iterations was performed on the ResNet-50 pretrained network, this was the suggested default of DeepLabCut. Furthermore upon analysis of novel videos, DeepLabCut would return a likelihood value indicating the confidence in the predicted location, this was a value between (0,1). An example of the typical output from DeepLabCut can be seen in Table 4.3, this example could be extended to include the two remaining markers. Each frame would have information about all three markers, even if the predicted likelihood is extremely low.

Table 4.3: Example of the data format that DeepLabCut returns for analyzed videos

frame	u	v	likelihood	label
0	125	600	0.978	WandTop
1	127	612	0.867	WandTop

4.2.2 DeepLabCut Iterations Overview

White Wand Markers

There were two iterations of CNNs used in the project. The first was trained on the initial calibration wand with only white markers. The performance of this network was not suitable for our applications as there was confusion between the markers. When a novel video was introduced to the CNN, the poor generalizations were made clear by markers being placed incorrectly or randomly. An example of the expected out put from the first iteration is shown below in Figure 4.3 Further analysis of this iteration will be

Figure 4.3: First wand iteration correctly labeled, Top(Blue), Middle(Green), Bottom(Red)

covered in the results and discussion section.

Colour Wand Markers

In the second iteration, a wand with coloured points was used in an attempt to improve the distinctness between markers, the final calibration wand can be seen in Figure 4.2. As alluded to earlier, the data set in the second iteration was augmented with addition GoPro footage to improve the robustness of the CNN. To improve the robustness, additional video footage was collect from varying scenes. The following list highlights the approach taken to improve the increase the variability.

1. Record at different times of day, this results in varying light levels and shadows.
2. Cover a large range of backgrounds with varied textures. i.e. a smooth wall or scene broken up by trees.
3. Capture footage indoors and outdoors. i.e. on a field or in an enclosed parking lot
4. Vary the motion and orientation of the wand when moving through the volume. i.e point at the camera and rotate the wand.

The additional footage was only recorded on the GoPro HS5s as they were planned to be used in the recording of the calibration footage for trajectory optimization. Lastly, Figure 4.4 shows the intended operation of the Colour Wand CNN in a setting with six cameras. **Please Note**, the colours that DeepLabCut used to label the marks changed

between iterations, the correct association is now: Top (Blue), Middle (Red) and Bottom (Green),

Figure 4.4: Second wand iteration correctly labeled in a 6 camera setting, Top(Blue), Middle(Red), Bottom(Green)

4.3 Trajectory Optimization

This section will detail the approach to the trajectory optimization problem and how it was set out. The general approach to the problem could be broken down into the following stepes:

1. Define the pose of the wand in general coordinates.
2. Formulate the required parameters for the trajectory problem.
3. Identify the constraints of the problem.
4. Created an objective function.
5. Initialize the problem.

Once again an iterative approach was taken towards the trajectory optimization section of the project. The goal was to develop the trajectory problem from a simple two camera case and scale it up towards six cameras as progress was made.

Some important notes and assumptions during for the trajectory optimization:

1. The intrinsic parameters are known
2. An initial estimate of the extrinsic parameters was known, this was used for initialization.
3. All the cameras were treated as identical and thus had the same intrinsic parameters, including distortion coefficients.

4.3.1 Wand Pose to Describe the Model

In order to solve the trajectory optimization problem, the wand pose had to be defined. Defining a pose model would allow the wand to be related to the inertial frame. The wand pose was chosen to be defined in terms of generalized coordinates and were as follows:

$$\mathbf{q} = [x, y, z, \phi, \psi]^T \tag{4.1}$$

Where the centeroid of WandTop was taken as the 3D point (x, y, z), denoted by P_{top}. Angles ϕ and ψ define the orientation of the calibration wand relative to the inertial frame. The generalized coordinates are shown in Figure 4.5.

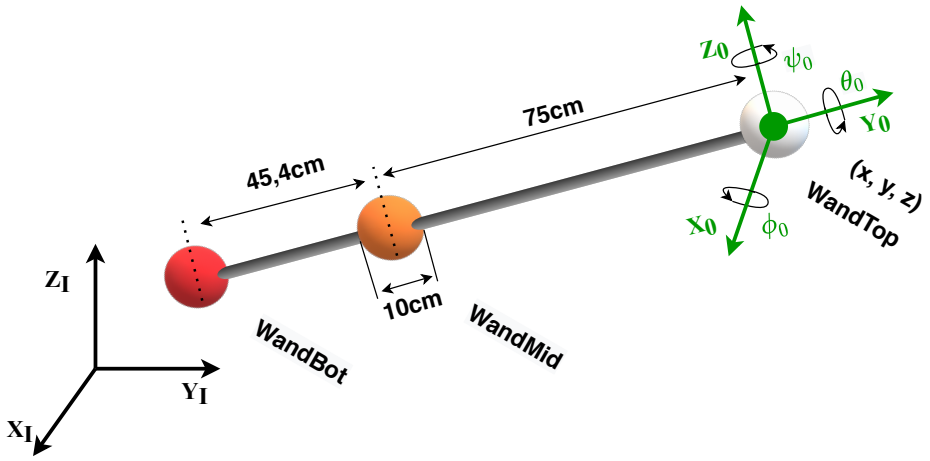


Figure 4.5: The wand model showing the chosen generalized coordinates and physical dimensions required to describe the remaining markers.

Before defining the points of WandMid and WandBot, the order of rotations must be defined. The order of rotations refers to which axis will be rotated about first by a given angle. It was decided that the order of rotation will follow the Euler 3-2-1 convention, this translates to a rotation about x-axis first, followed about the y-axis and concluding with the z-axis. This is shown below:

$$\mathbf{R}_0^I = \mathbf{R}_z(\psi) \times \mathbf{R}_x(\phi) \tag{4.2}$$

The rotation, $\mathbf{R}_y(\theta)$, is omitted since any rotation about the y-axis has no significant affect on the orientation of the wand. It is for this same reason that θ does not form part of the generalized coordinates.

Given the order of rotations and the generalized coordinates, the positions of WandMid, P_{mid}, and WandBot, P_{bot}, can now be defined.

$$P_{top} = [x, y, z]^T \tag{4.3}$$

$$P_{mid} = P_{top} + \mathbf{R}_I^0[0, -0.75, 0]^T \tag{4.4}$$

$$P_{bot} = P_{top} + \mathbf{R}_I^0[0, -1.204, 0]^T \tag{4.5}$$

From the pose model, it was clear that there should be 5 pose parameters to fully describe the entire wand in any orientation.

4.3.2 Trajectory Parameters and Variables

Before designing the structure of the optimization problem it is beneficial to identify the parameters and variables that are part of the model that will be optimized in the trajectory problem. Parameters can be thought of as known values in a trajectory problem and the variables are the values that the optimization actually solves for.

Beginning with the parameters, a list of defined parameters are shown in Table 4.4 below.

Table 4.4: Description of the parameters defined for the wand model

Parameter	Description
n	Number of nodes or steps in the trajectory
p	Number of pose parameters
m	Number of markers on the wand
c	Number of cameras
h	The time step between points of the trajectory

In particular, p, refers to the number of parameters required to define the pose of the wand. In this case, the wand pose can be defined by just the generalized coordinates.

The variables that were required for this trajectory problem were identified as follows:

1. Variables to define the position, velocity and acceleration of the trajectory.
2. One to describe the 3D poses of each marker.
3. To describe a rotation matrix

4. To describe the translation vector

The trajectory position of the wand was defined by a variable, $\mathbf{x}$. It was made up of the pose parameters, p, at each of the trajectory steps, n. Given that $\mathbf{x}$ is the position of the trajectory, the first and second time derivative could be take to obtain the "velocity", $\dot{\mathbf{x}}$, and "acceleration", $\ddot{\mathbf{x}}$, trajectory variables respectively.

Each camera had a rotation and translation variable. The camera pose variables, $\mathbf{R}$ and $\mathbf{t}$, were each made up of three values. The rotation matrices were converted to Rodrigues rotation angles as the rotation matrix could be represented by three values instead of nine. This reduced the degrees of freedom for the optimization problem and allowed it converge to a sensible solution. The process to convert between Rotation matrices and Rodrigues angles are shown below in equation (4.6) [1].

$$\begin{aligned} &\text{Given a Rodrigues vector with 3 angles,} \\ &r = [r_1, r_2, r_3]^T \\ &\theta = \sqrt{r_1^2 + r_2^2 + r_3^2} \\ &r \leftarrow r/\theta = [r_x, r_y, r_z]^T \\ &\mathbf{R} = \cos(\theta) I + (1 - \cos\theta) r r^T + \sin(\theta) \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \end{aligned} \tag{4.6}$$

Where I is a 3×3 identity matrix.

The variables defined for the optimization problem are shown below in Table 4.5.

Table 4.5: Trajectory Variables associated with the wand model

Variable	size	Description
$\mathbf{x}$	$n \times p$	Position at each node
$\dot{\mathbf{x}}$	$n \times p$	Velocity at each node
$\ddot{\mathbf{x}}$	$n \times p$	Acceleration at each node
Rots	$c \times 3$	Rodrigues Rotation angles
Trans	$c \times 3$	postions
Q	$n \times m \times 3$	The 3D points of the markers
$\mathbf{w}$	$n \times p - 1$	Model acceleration error
$\mathbf{v}$	$n \times c \times m \times 2$	2D measurement error

Lastly, variables $\mathbf{v}$ and $\mathbf{w}$ were introduced to cater for residual error. In the case of the measurement error, $\mathbf{v}$, was defined as the residual reprojection error, i.e. the difference between the estimate points and the measured points. The model acceleration error, $\mathbf{w}$, was intended to define the acceleration error of each marker between frames. The reason being that the acceleration between frames was assumed to be constant. However, realis-

tically this is not the case and there would be some difference between the accelerations of the separate frames [4].

4.3.3 Constraints

Constraints are used to limit the trajectory optimization to operate within the bounds of the model being defined. There were two main types of constraint introduced for the wand trajectory optimization problem and they are based off Clarks implementation [4].

- Model Constraints and,
- Measurement Constraints

Model Constraint

The model constraint was set up in a way to define some form of model dynamics between steps in the trajectory. It was intended to act as a crude estimate of the system dynamics for the wand model. It was developed in precisely the same way as Clark [4] and was based on the assumption of constant acceleration between trajectory steps. In order to account for the definite errors that would occur, the model acceleration error, $\mathbf{w}$, was included within the model constraint.

The model constraint was created in the following way using implicit Euler integration to relate the model variables: $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ to each other [4].

$$\mathbf{x}_i = \mathbf{x}_{i-1} + h\dot{\mathbf{x}}_i \tag{4.7}$$

$$\dot{\mathbf{x}}_i = \dot{\mathbf{x}}_{i-1} + h\ddot{\mathbf{x}}_i \tag{4.8}$$

$$\ddot{\mathbf{x}}_i = \ddot{\mathbf{x}}_{i-1} + \mathbf{w}_i \tag{4.9}$$

Where $i = [2, ..., n]$ and h is the time step between trajectory steps.

Measurement Constraint

The measurement constraints were also based off Clarks work [4], save for the addition of the rotation, **Rotz**, and translation, **Trans**, variables into the constraint definition. The measurement constraint was defined with the intention of relating estimated 3D poses

of the wand markers to the pixel data retrieved from DeepLabCut. This was done by repojecting the 3D estimate poses to pixel coordinates for each camera. The measurement error, **v**, was included to account for the difference between the estimated points and the data retrieved from DeepLabCut.

In Clarks work [4], the intrinsic and extrinsic parameters of the camera were set to be constants within this constraint and thus only the estimated 3D points would be adjusted as a result. The addition in this study was to include the extrinsic camera parameters as variables within this constraint. Thus as the new constraint was enforced the 3D estimate points and the extrinsic parameters would both be optimized for all the cameras.

The constraint took the form:

$$\hat{x}_{i,j} - x_{i,j} - v_{i,j} = 0 \quad \text{for } i = (1, 2, ..., n), j = (1, 2, .., c) \tag{4.10}$$

Where $\hat{x}$ is the estimated pixel points and x, is the DeepLabCut pixel point.

Two additional constraints were added for the rotation and translation variables, **Rots** and **Trans**. These constraints intended to ensure the optimizer produced a single solution for each camera pose over the entire trajectory.

$$\mathbf{Rots}_i = \mathbf{Rots}_{i+1} \quad \text{for } i = (1, 2, ..., n) \tag{4.11}$$

$$\mathbf{Trans}_i = \mathbf{Trans}_{i+1} \quad \text{for } i = (1, 2, ..., n) \tag{4.12}$$

4.3.4 Objective Function and Initialization

Objective Function

The objective function was based off the work of Clark [4] and was composed of two sub-cost functions, one relating to the acceleration error and the other to the measurement error. Unfortunately due to time constraints, implementation and testing was only conducted based on an objective function associated with measurement loss and no dynamics. Thus it will be the only function defined here.

The objective function aimed to minimize the measurement error and was based off a redescending loss function. The choice of redescending loss function was intended to reduce the impact of outliers on the minimization. For instance, if the residual error

surpasses a threshold value, the cost associated with that error no longer increases. Thus the cost associated with an error which is just above the threshold, i.e a "poor" estimate, has the same influence as an outlier error with ten times the magnitude.

Note in the objective function, the measurement error was normalized by dividing by the standard deviation σ as Clark [4] performed. The value was normalized in anticipation for the addition of the model dynamics so both terms would be weighted evenly. Unfortunately that stage was not reached.

The objective function was defined as follows,

$$\min_{\mathbf{x},\mathbf{Q},\mathbf{Rots},\mathbf{Trans},\mathbf{v}} \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{m}\sum_{m=1}^{2} C(\frac{v_{i,j,k,m}}{\sigma}) \tag{4.13}$$

Where C(.), was the redescending loss function defined by the following equation [4]:

$$C(e) = \begin{cases} \frac{e^2}{2} & |e| \leq a \\ a|e| - \frac{a^2}{2} & |e| \geq a, |e| < b \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2}(1 - (\frac{c-|e|}{c-b})^2) & |e|, |e| < c \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2} & |e| \geq c \end{cases} \tag{4.14}$$

Where $a = 2, b = 4, c = 10$

In explanation of the objective function, the measurement error was minimized for each 2D marker point, for each marker, per camera along the entire trajectory.

Initialization

The trajectory optimization was initialized from estimated 3D points based on data taken from DeepLabCut which had a likelihood of greater than 0.5. The 3D estimate points were calculated with OpenCV's triangulation function, which also took in the initial estimates for the intrinsic and extrinsic parameters.

Notably, the initial values for the extrinsic parameters were taken to be the result of the checkerboard extrinsic calibration, before the sparse bundle adjustment.

Chapter 5

Results

This section serves as a means to represent the collected results from the design phase. It particularly highlights the results from the training of DeepLabCut and the Trajectory Optimization Problem.

5.1 DeepLabCut Performance Results

5.1.1 White Wand Results

Upon evaluating the performance of the iteration of DeepLabCut trained on the calibration wand with white markers, the following statistics were collected and are displayed in Table 5.1

Table 5.1: DeepLabCut evaluation results returning the reprojection error Means in pixels for all points and only those above the cut-off likelihood, Wand white iteration

	Mean	Mean After Cut-off	Likelihood Cutoff
Train Error	4.08	3.48	0.6
Test Error	36.93	32.09	0.6

The training error residuals were plotted after removing extremely large outliers, the resulting histogram was as follows:

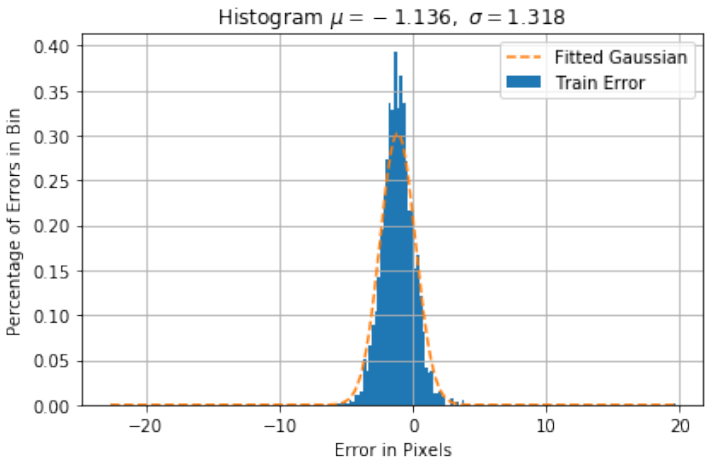


Figure 5.1: A Histogram showing the residual error from the training data set and fitted Gaussian curve after large outliers had been removed.

Further investigation of the performance was done by analysing novel videos to gain a visual idea of the performance, Figures 5.2 and 5.3 were extracted from an analysed video.

Figure 5.2: First wand iteration correctly labeled, Top(Blue), Middle(Green), Bottom(Red)

The network was expected to label WandTop as blue, WandMid as green and WandBot as Red. It is evident from Figure 5.3 that the network needed to be retrained.

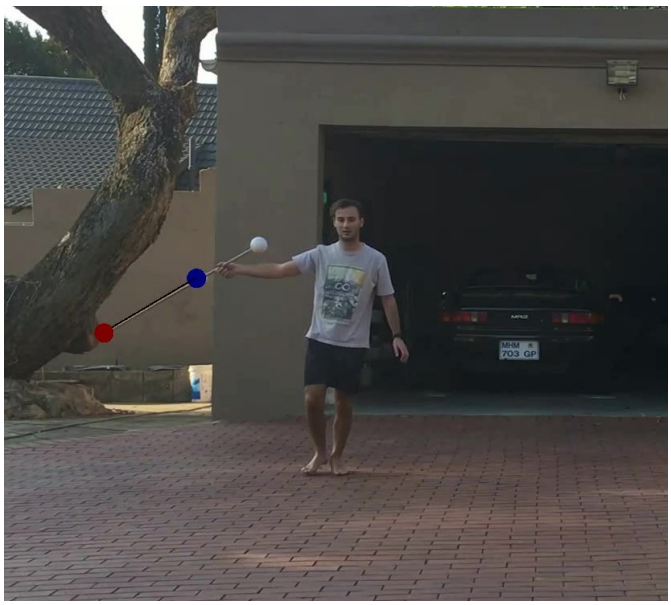


Figure 5.3: First wand iteration incorrectly labeled, The middle is labeled as the top and the top is labeled as the bottom.

From the analysed video, the range of likelihoods were plotted, this figure is large and can be found in Appendix A as Figure A.1.

5.1.2 Colour Wand Results

The colour wand was neural network was evaluated using DeepLabCut's provide functionality, the resulting statistics for the mean reprojection error were as follows:

Table 5.2: DeepLabCut evaluation results returning the reprojection error Means in pixels for all points above the cut-off likelihood, Wand colour iteration

	Mean	Mean After Cut-off	Likelihood Cutoff
Train Error	2.74	2.74	0.5
Test Error	17.3	10.7	0.5

A Histogram of the training residual error, the data was filtered of extremely large outliers before plotting. The result was as follows.

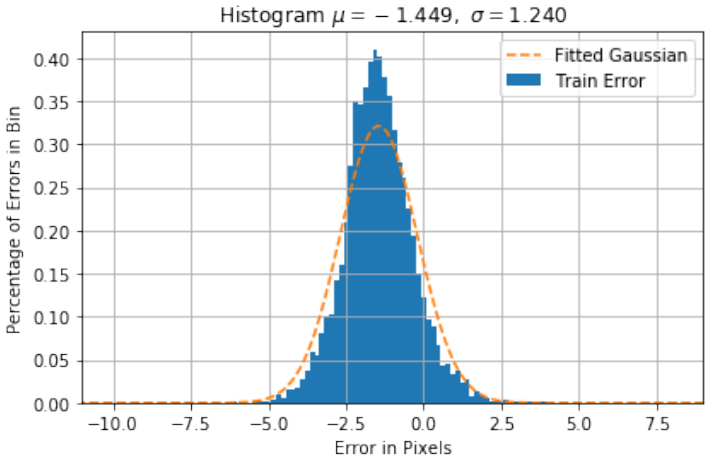


Figure 5.4: A Histogram showing the residual error from the training data set and fitted Gaussian curve after large outliers had been removed.

5.2 Trajectory Optimization Results

Due to the lack of meaningful results and the significant differences between the checkerboard calibration and the wand calibration results, it was decided that the Rodrigues forms of the Rotation matrices and the translation vectors could be presented for a direct comparison.

5.2.1 Results For 2 Cameras

The trajectory optimization solving the case of two cameras in order refine the extrinsic parameters.

Table 5.3: Rodrigues Angles and Translation vectors for 2 Camera setup, comparing the difference between the Wand and Checkerboard calibration

	R_1	R_2	t_1	t_2
Wand	$\begin{bmatrix} 1.4090 \\ -0.2491 \\ -0.1023 \end{bmatrix}$	$\begin{bmatrix} 1.3926 \\ -0.5298 \\ 0.2909 \end{bmatrix}$	$\begin{bmatrix} -0.0093 \\ 0.0355 \\ -0.02835 \end{bmatrix}$	$\begin{bmatrix} 6.1026 \\ -0.4956 \\ -1.4385 \end{bmatrix}$
Checkerboard	$\begin{bmatrix} 1.5708 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1.538308 \\ -0.2459 \\ 0.4559 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 6.127 \\ -0.2490 \\ -1.4597 \end{bmatrix}$

The visualization of the data from Table 5.3

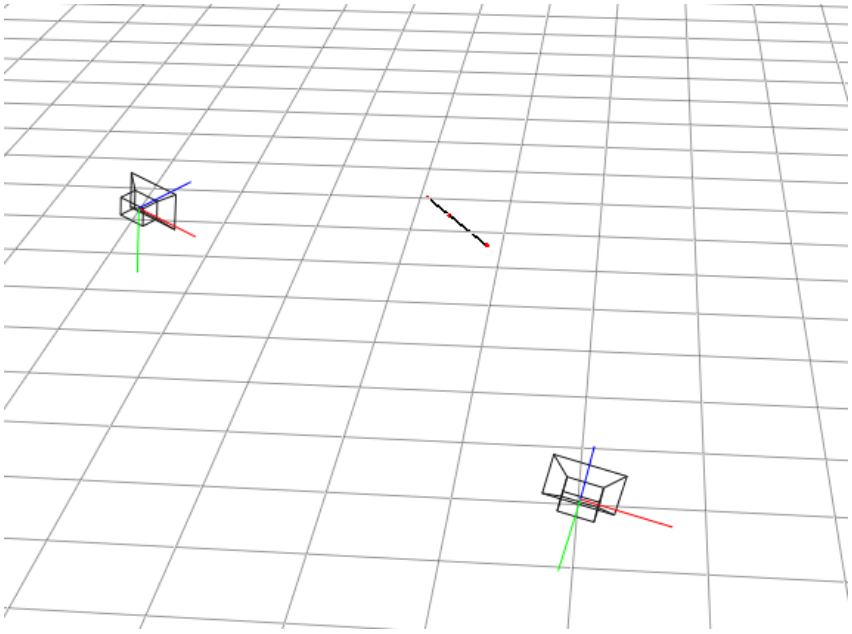


Figure 5.5: Visualization of the extrinsic calibration results for a 2 camera case using trajectory optimization

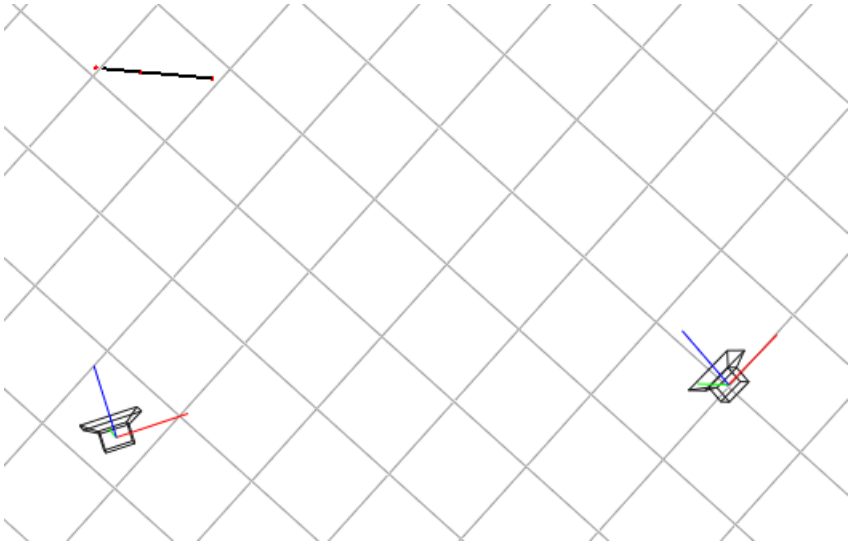


Figure 5.6: Visualization of the extrinsic calibration results for a 2 camera case using trajectory optimization, Top view

5.2.2 Results For 3 Cameras

Trajectory optimization solving the case of three cameras and refining the extrinsic parameters.

Table 5.4: Rodrigues Angles and Translation vectors for 3 Camera setup, comparing the difference between the Wand and Checkerboard calibration

	R_1	R_2	R_3	t_1	t_2	t_3
Wand	$\begin{bmatrix} 1.4344 \\ -0.0597 \\ -0.0189 \end{bmatrix}$	$\begin{bmatrix} 1.2749 \\ -0.2210 \\ 0.6203 \end{bmatrix}$	$\begin{bmatrix} 0.7436 \\ -0.4382 \\ 0.9607 \end{bmatrix}$	$\begin{bmatrix} -0.0487 \\ 0.0333 \\ 0.0317 \end{bmatrix}$	$\begin{bmatrix} 6.3465 \\ -0.0945 \\ -0.6559 \end{bmatrix}$	$\begin{bmatrix} 9.3401 \\ 1.8567 \\ 10.8399 \end{bmatrix}$
Checkerboard	$\begin{bmatrix} 1.5708 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1.538308 \\ -0.2459 \\ 0.4559 \end{bmatrix}$	$\begin{bmatrix} 1.1230 \\ -1.0214 \\ 1.5018 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 6.127 \\ -0.2490 \\ -1.4597 \end{bmatrix}$	$\begin{bmatrix} 7.5049 \\ 1.5248 \\ 10.3374 \end{bmatrix}$

The visualization of the data from Table 5.4

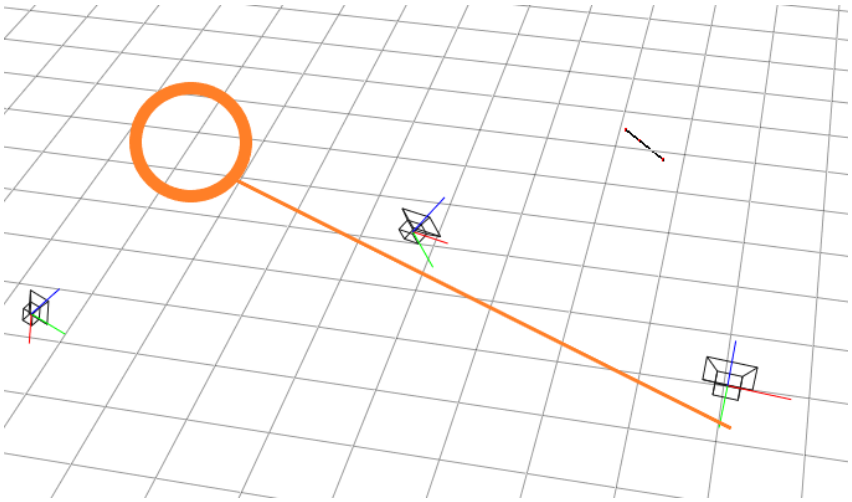


Figure 5.7: Visualization of the extrinsic calibration results for a 3 camera case using trajectory optimization, the orange circle highlights the expected region for the third camera to appear as the cameras were placed in a straight line in practice.

5.2.3 Results For 6 Cameras

The trajectory optimization solving for a six camera case in order to determine the extrinsic parameters. The resulting placement of the cameras for illustration purposes.

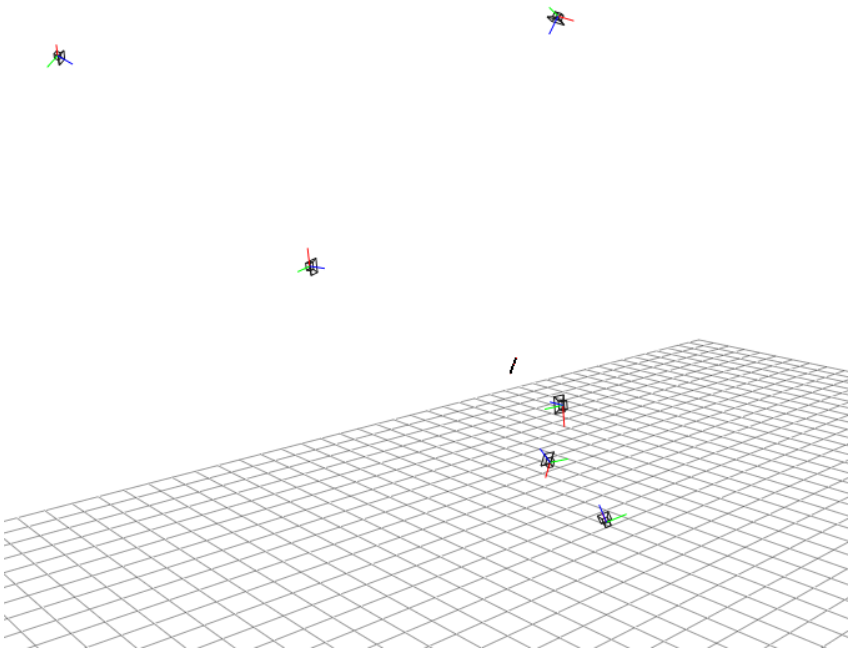


Figure 5.8: The resulting 6 camera orientation relative to each other and the wand, given the extrinsic calibration from the trajectory optimization.

Chapter 6

Discussion

The discussion of the results will firstly be broken down into two parts. The first half will compare the two iterations of neural network and highlight the key factors that create the differences. The second part of this discussion will make mention of the trajectory optimisation problem, its performance and possible areas which caused such poor results.

For ease of reading iteration one will refer to the case where the wand was composed of white markers and iteration two will refer to the case where two markers were coloured. As alluded to in the design and now made clearer from the results section, the first iteration of the neural network had poor performance. It was particularly bad at distinguishing the different marker points as can be seen in Figures 5.3 and 5.2. From this result however, it was realised that the initial difference of biasing the middle marker to one side did not work and no feature detector was developed for this difference.

A further telling sign that the network did not develop well was evident from the test error Mean in Table 5.1. Although the mean is a statistic that is easily influenced by outliers it is still relatively high (approximately 20 pixels) in comparison to the test error Means for the second iteration, shown in Table 5.2. This suggests that the first iteration did not have suitable feature detectors to consistently identify the same points on the wand in its initial form.

Observing the results of the second iteration its is clear that DeepLabCut developed more suitable means to identify the markers when colour was added to the wand. The lower Means errors indicate that the neural network was more consistent at labeling data with coloured markers. In addition, it could be inferred that fewer outliers were present due to the improved feature detectors developed the second time. A point however, should be made that there was almost twice the amount of labeled data used in the second iteration in comparison to the first. Lastly, both networks were only trained for 100 000 iterations during the training process. The reason the networks were trained for such as short time, was a precaution to prevent possible over-fitting of the network to the small data set. Both of these reasons may be additional points why the two networks differed in performance.

Transitioning to discussing the two calibration approaches. From the Methods section it was clear that the checkerboard provided a good initial estimate and optimized set of parameters from the mean and standard deviation of the reporjection errors being well below zero. A note on the checkerboard calibration, it was notice that the standard

deviation and mean reprojection errors increased slightly after the SBA. This could possibly be attributed to a slightly poor initialization resulting the optimizer locating a local minimum which was greater than the start. The the resulting extrinsic parameters from checkerboard calibration were plotted in Figure 6.1 to illustrate what the expected result of the calibration process.

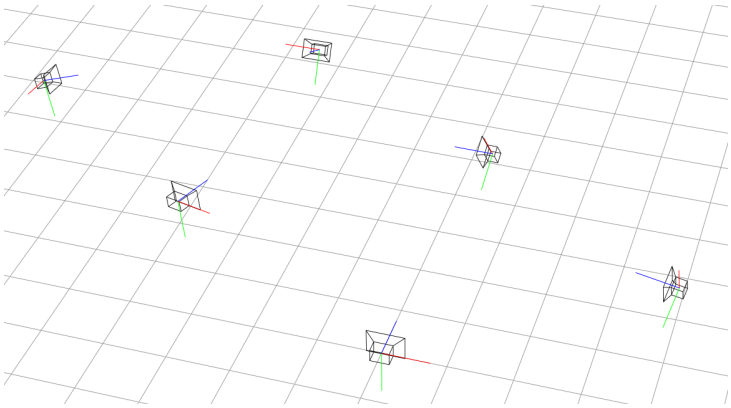


Figure 6.1: This figure illustrates the expected out come of the orientations and positions of the cameras. This was plotted using the checkerboard extrinsic parameters.

In comparison to the trajectory optimization technique employed, the checkerboard calibration outclassed it and it is evident why a 2D pattern can be used for a ground truth. However, there is still merit in the calibration technique associated with a wand and of course using trajectory optimization. From the results of the 2 camera test in Table 5.3, the resulting Rodigues angles from the trajectory optimization were a maximum of 0.28 units off the checkerboard Rodrigues angles. The values for the translations were similarly as close to those of the checkerboard calibration.

Unfortunately the seemingly "good" results viewed from the 2 camera test did not translate well to the 3 camera and 6 camera case. Starting with the 6 camera case, it was clear from the resulting 3D visualizations of the cameras in Figure 5.8 that there was fundamental problem in the definition of the trajectory optimization problem. After consideration and analysis of the results, it was thought that the issue may relate to the values used to define the shape of the redescending loss function and the value of sigma used to normalize the measurement error. The reason being, as the values were adjust the results would seem to converge slowly towards the solutions of the Checkerboard. This approach was abandoned unfortunately due to time constraints and lack of reason to be adjusting the values of the loss function. This behaviour was particularly evident with the 3 camera case and no such behaviour was observed for the 6 camera case.

The trajectory optimization technique has by no means performed well, however there is definitely room for improvement and could well provide solutions of the same caliber as the checkerboard calibration.

Chapter 7

Conclusions

The aim of the project was to implement a way to automatically determine the extrinsic camera parameters in a multi-camera arrangement. The proposed method was to use a 1D calibration wand, whose position markers were automatically identified and tracked by DeepLabCut. The extracted points were then used in a trajectory optimization algorithm to refine the extrinsic parameters.

Unfortunately the proposed method did not work sufficiently. In the case of two cameras being calibrated to each other, the trajectory optimization produces some comparable results to that of the ground truth checkerboard method. In the subsequent iterations of three and six cameras, neither produced results which could be used as an extrinsic calibration.

With regards to tracking the calibration wand, the second iteration of the neural network developed some good feature detection capability and returned the locations of the 2D markers to an adequate standard such that the data could be used in a calibration process.

The calibration wand was a suitable object to perform the process given it characteristics which are favoured for multi-camera arrangements. It was evident that there was some fundamental issues associated with the trajectory optimization problem, however there was no further time to explore and further diagnose the cause of the issue.

Lastly there is still plenty of opportunity for the automatic calibration to be achieve with a 1D pattern as the developed process can be built upon given that its relative achievement in a 2 camera case.

Chapter 8

Recommendations for Future Work

There is still suitable room to improve the project and develop into a form which can be used to simplify camera calibration in a small lab. The following serves as some recommendations for future work:

1. An improved method should be explored for identifying the center of the markers on the calibration wand.

The problem observed in this project was that, DeepLabCut would variably label the marker in an analyzed video. For instance, it would with no guarantee return the center of the marker. Thus introducing variability into calibration process as the trajectory optimization problem treated the identified points as the centroid of the marker, which was rarely the case. Forbes [24] proposes a method for locating the center of reprojected circles in order to correctly label the same location each time. This was not in the scope of the project but could be beneficial.

Further improvement could be made on simple aspects such as:

1. A more accurately fabricated calibration object would like improve the accuracy. The method in this project proved more to be a proof of concept which could be later improved if required.

Bibliography

[1] O. D. Team, "Camera calibration and 3d reconstruction," Dec 2019. [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=stereoCalibrate

[2] "Camera calibration pattern generator." [Online]. Available: https://calib.io/pages/camera-calibration-pattern-generator

[3] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "Using deeplabcut for 3d markerless pose estimation across species and behaviors," *Nature protocols*, vol. 14, no. 7, pp. 2152–2176, 2019.

[4] L. J. Clark, "Markerless 3D Motion Capture of Cheetahs in the Wild," 2020.

[5] H. Liu, "Face technologies on mobile devices," in *Facial Detection and Recognition on Mobile Devices*, H. Liu, Ed. Edinburgh: Morgan Kaufmann, 2015, pp. 11 – 38. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124170452000021

[6] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers," in *2015 12th Conference on Computer and Robot Vision*. IEEE, 2015, pp. 62–69.

[7] "Computer vision: What it is and why it matters." [Online]. Available: https://www.sas.com/en_us/insights/analytics/computer-vision.html

[8] E. DAVIES, "Chapter 21 - image transformations and camera calibration," in *Machine Vision (Third Edition)*, third edition ed., ser. Signal Processing and its Applications, E. DAVIES, Ed. Burlington: Morgan Kaufmann, 2005, pp. 595 – 623. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780122060939500241

[9] F. Remondino and C. Fraser, "Digital camera calibration methods: considerations and comparisons," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 266–272, 2006.

[10] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[11] Oct 2019. [Online]. Available: https://www.vicon.com/hardware/devices/calibration/

[12] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[13] MathWorks, "What is camera calibration?" [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[14] Imatest, "Projective camera model." [Online]. Available: https://www.imatest.com/support/docs/pre-5-2/geometric-calibration/projective-camera/

[15] S. H. Ahn, "Homogeneous coordinates," 2005. [Online]. Available: http://www.songho.ca/math/homogeneous/homogeneous.html

[16] Imatest, "Homogeneous coordinates," 2020. [Online]. Available: https://www.imatest.com/support/docs/pre-5-2/geometric-calibration/homogeneous-coordinates/

[17] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[18] Q. Fu, Q. Quan, and K.-Y. Cai, "Calibration of multiple fish-eye cameras using a wand," *IET Computer Vision*, vol. 9, no. 3, pp. 378–389, 2014.

[19] K. K. Delibasis and I. Maglogiannis, "Feature extraction and pattern recognition from fisheye images in the spatial domain." in *VISIGRAPP (4: VISAPP)*, 2018, pp. 460–465.

[20] G. Xu, A. Zheng, X. Li, and J. Su, "A method to calibrate a camera using perpendicularity of 2d lines in the target observations," *Scientific reports*, vol. 6, p. 34951, 2016.

[21] Z. Zhang, "Camera calibration with one-dimensional objects," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 7, pp. 892–899, 2004.

[22] "Camera calibrator." [Online]. Available: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html

[23] T. Pribanić, P. Sturm, and S. Peharec, "Wand-based calibration of 3d kinematic system," *IET computer vision*, vol. 3, no. 3, pp. 124–129, 2009.

[24] K. Forbes, A. Voigt, and N. Bodika, "An inexpensive, automatic and accurate camera calibration method," in *Proceedings of the Thirteenth Annual South African Workshop on Pattern Recognition*, 2002, pp. 1–6.

[25] J. A. De FrançA, M. R. Stemmer, M. B. d. M. França, and J. C. Piai, "A new robust algorithmic for multi-camera calibration with a 1d object under general motions without prior knowledge of any camera intrinsic parameter," *Pattern Recognition*, vol. 45, no. 10, pp. 3636–3647, 2012.

[26] J. Mitchelson and A. Hilton, "Wand-based multiple camera studio calibration," *Center Vision, Speech and Signal Process*, 2003.

[27] R. I. Hartley, R. Gupta, and T. Chang, "Stereo from uncalibrated cameras." in *CVPR*, vol. 92, 1992, pp. 761–764.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[29] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[30] M. P. Kelly, "Transcription methods for trajectory optimization: a beginners tutorial," *arXiv preprint arXiv:1707.00284*, 2017.

[31] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[32] T. Hedrick, "Easywand and easycamera stereo videography software." [Online]. Available: http://www.cs.bu.edu/faculty/betke/jeb/

Appendix A

Additional Information

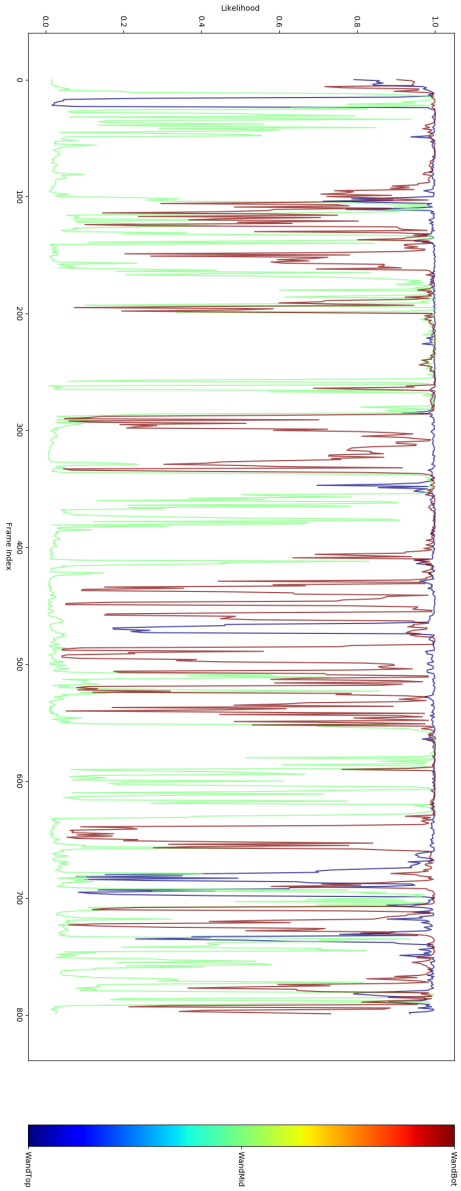


Figure A.1: Video analyzed using the first iteration of DeepLabCut, The likelihood of the middle marker(green) is distinctly low for the majority of the video. The wand was in frame for the entire video.

ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Mauro Borrageiro
Department		Electrical Engineering
Preferred email address of applicant:		BRRMAU002@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc(Eng) Mechatronics
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Dr. Amir Patel
If this is a researchcontract, indicate the source of funding/sponsorship		NA
Project Title		Automatic Multi-Camera Extrinsic Calibration in the Wild

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Mauro Giovanni Borrageiro		15/08/2020
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Amir Patel		18/08/2020

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).			
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			