UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Extrinsic calibration of multiple partially overlapping RGB-D cameras

Capstone Thesis

Huy Nhat Minh Nguyen

Student Number: 13734569

Project Number: AUT-22-07822

Major: Mechatronics Engineering

Supervisor: Dr. Alen Alempijevic

Co-Adviser: Mr Jasprabhjit Mehami

41030 - Engineering Capstone

Bachelor of Engineering (Honours)

University of Technology Sydney

Faculty of Engineering and Information Technology

31 May, 2022

Author's declaration

I, *Huy Nhat Minh Nguyen* declare that this thesis, submitted in fulfilment of the requirements for the Bachelor of Engineering (Honours) in the *School of Mechanical and Mechatronic Engineering*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney, Australia, is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

SIGNATURE: _____

[Minh Nguyen]

DATE: 30$^{\text{th}}$ May, 2022

PLACE: Sydney, Australia

Abstract

Exteroceptive sensors and cameras have been widely implemented in various engineering fields, especially in robotics and vision. Working with a camera or a multiple camera system requires accurate camera parameters, including the intrinsic and extrinsic parameters. This led to the introduction of different camera calibration methods with varying calibration targets, depending on specific camera types and project purposes.

Planar calibration targets have been extensively applied in multiple camera system calibration methods due to their simple implementation, high precision and cheap production, compared with 3D calibration targets. However, in working with calibrating a system with numerous cameras, some planar calibration targets reveal certain inconveniences, especially for extrinsic calibration. For example, in a system with multiple cameras facing each other, all cameras cannot completely capture the planar calibration target. This limits the application of the calibration method using planar calibration target to various camera system configurations.

This capstone project proposes a method using an augmented planar calibration target to calibrate a multiple partially overlapping RGB-D camera system. A double-sided planar board is employed to avoid the limitations of planar targets, and the ArUco marker system is adopted as fiducial marker to support the detection algorithm. In addition, this calibration method applies the General Graph Optimisation (G2O) framework to optimise and guarantee the accuracy of the extrinsic parameters during the calibration process. This project experimentally validates the method's accuracy and discusses the applicability of the proposed technique. Finally, this project provides the framework's source code and supporting documentation, which is mainly programmed in C++ and MATLAB with ROS as a middle-ware, for further use and development.

Acknowledgments

This capstone project as well as the thesis would not be successful without the instruction and assistance of the people around me. I would particularly like to acknowledge:

Special thanks to Dr Alen Alempijevic, my supervisor, for being supervised this capstone project. His mentorship and guidance is the role key of the success of this project. It is my honour to have the opportunity to work with him.

Mr Jasprabhjit Mehami, for following me closely and helping me throughout the process of working on the project. His knowledge and experience in camera and vision contributed greatly to the success of this approach.

My family and loved ones, for being always by my side to support me mentally, helping me to always feel comfortable and confident to complete this capstone project as well as the thesis.

My friends in Australia, for their support whenever I encountered a struggle during the process of doing this capstone project.

Contents

List of Tables vii

List of Figures ix

1 Introduction 1

1.1 Project background . . . 1

1.2 Project scope . . . 2

1.3 Thesis outline . . . 3

2 Literature Review 4

2.1 Camera calibration . . . 4

2.2 Multi-camera calibration . . . 5

2.2.1 Stereo cameras . . . 5

2.2.2 Non-overlapping Field of View cameras . . . 5

2.3 Fiducial markers . . . 10

2.4 General Graph Optimisation . . . 11

3 Approach 12

3.1 Camera model . . . 13

3.2 Calibration target configuration: double-sided ArUco board . . . 16

3.3 Pose Graph Optimisation . . . 17

4 Evaluation and Results 18

4.1 Overview . . . 19

4.2 Camera system . . . 20

4.3 Double-sided ArUco board configuration . . . 22

4.4 Board pose estimation 27
4.5 Pose Graph Optimisation 31

5 Discussion **36**
5.1 Accuracy 36
5.2 Further works 38

6 Conclusion **40**

Bibliography **41**

List of Tables

4.1 Intrinsic parameters of each camera. 21
4.2 Double-sided ArUco board configuration 25
4.3 Pose Graph Optimisation result 34

List of Figures

2.1 Ueshiba & Tomita (2003) calibration method for overlapping camera system . . . 6
2.2 Dong et al. (2016) method for non-overlapping multiple camera system 6
2.3 Double-sided checkerboard of Marcon et al. (2017) . . . 7
2.4 Yin et al. (2018) calibration method . . . 8
2.5 Spherical calibration targets of Shen & Hornsey (2011) . . . 9
2.6 Fiducial markers . . . 10
2.7 Multiple camera system on a legged robot . . . 11

3.1 Perspective-n-Point (PnP) pose problem . . . 14
3.2 ArUco double-sided board configuration . . . 16
3.3 Pose Graph represents the calibration method . . . 17

4.1 RGB-D Camera Intel Realsense D435i . . . 20
4.2 Camera layout . . . 21
4.3 Double-sided ArUco board (front side) . . . 23
4.4 Double-sided ArUco board (back side) . . . 23
4.5 SolvePnP() function in Open-CV library in detail . . . 27
4.6 ArUco markers detection in image . . . 28
4.7 ArUco board estimation (front side) . . . 29
4.8 ArUco board estimation (back side) . . . 30
4.9 Nodes CSV file . . . 32
4.10 Edges CSV file . . . 33

4.11 Poses of the optimised cameras and calibration board. Cam 7, which is coloured in red, is the origin. Every 10^{th} pose of the calibration board's origin represented by a 3D axis 34

4.12 The pose graph before and after the optimisation is carried out. Blue circles are the nodes, red lines are the edges, and blue lines are loop-closure edges. The four camera nodes have been labelled in blue. . . . 35

5.1 ArUco board estimation (both sides) 37

5.2 Edge Residual Errors by Edge ID 37

Chapter 1

Introduction

1.1 Project background

This thesis was completed in synergy with the "Objective real-time live cattle assessments to improve profit" funded project by Meat and Livestock Australia. The overarching project aims to develop and improve profitability and productivity of the beef value chain, addressing cattle non-compliance with processor grid specifications – a $51M pa issue for the sector and more when feeding costs are taken into consideration. This problem is aggravated by the failure to improve production efficiencies because data on cattle traits is unavailable in real-time. The industry opportunity is to have an objective real-time assessment of live cattle to assist producers to achieve greater production efficiencies by meeting "market specifications" and increasing lean meat yield so that additional dollars flow into producers' pockets.

The project has developed bespoke designed inserts for a race that ensure cattle can move freely, is safe and provides unobstructed views of cattle from multiple RGB-D cameras. We have created a software framework that enables measuring hip height and estimating P8 fat and Muscle Score. To effectively perform this software framework, a thoroughly calibrated 3D multiple-cameras system is strongly required. A calibrated multiple-cameras system would provide specific parameters of cameras to operate involved tasks.

Camera parameters included intrinsic parameters and extrinsic parameters, combined to define the mathematical relationship between 3D global points $P(X, Y, Z)$

and their corresponding 2D image plane points $p(u, v)$. The intrinsic parameters, also known as the camera matrix, belong to the camera itself, consisting of the focal length f, the principal point (the optical centre), and the lens distortion. Each camera has unique intrinsic parameters that come from the physical design process. The extrinsic parameters, or the camera pose, are used to illustrate the geometrical relationship (rotation and translation) between the camera itself and a feature in its external world (Ikeuchi, 2014). Specifically, in this project's scope, the extrinsic parameters of a multiple-camera system are the transformation relationship between each camera and others in this structure.

Nowadays, camera calibration is essential for any computer vision system. Several software toolboxes and packages are provided and widely adopted to simplify the calibration process as well as ensure the accuracy of the results, the most common being the Open-CV library or the MATLAB Computer Vision Toolbox. However, in particular multiple-camera systems, such as the multiple partially overlapping cameras system, conventional methods cannot be used for accurate parameter results.

This capstone report focus on researching and implementing a method for multiple partially overlapping RGB-D cameras calibration compatible with the ROS framework in C++ programming language. The approach is expected to solve the problem of extrinsically calibrating the camera system with fields of view overlapping and facing each other that cannot be handled using standard calibration methods.

1.2 Project scope

This capstone project proposes to use a double-sided ArUco board as the calibration target, with the support of the Open-CV library and the General Graph Optimisation (G2O) framework (Kümmerle et al., 2011). The software included with this method is implemented mainly in C++ and MATLAB, and ROS (Robot Operating System) is used as a middle-ware to transfer data between the program and the multiple-camera system.

1.3 Thesis outline

This capstone thesis is structured as follows:

Chapter 2 looks at the literature review of the previous camera calibration relevant to this project's purpose. This chapter evaluates the advantages and drawbacks of these methods, as well as considers the possibility and suitably to the project's objectives. It also explains why specific project decisions were taken.

Chapter 3 presents the approach and the methodology of this project. These ideas are described and explained in detail. This chapter also provides important insights necessary to understand the work of the project.

Chapter 4 demonstrates the implemented experiments to check the possibility and efficiency of the approach. The implementation, relevant equipment and results obtained are described in detail.

Chapter 5 discusses the methodology as well as the experiments implemented. This chapter reviews the accuracy of experimental results, clarifies the uncertainties and difficulties of this approach, and plans potential future works to improve this method.

Chapter 2

Literature Review

With the development of science and technology, computer vision has gradually become popular in various engineering fields, leading the demand for exteroceptive sensors to increase in many different applications. In order to work with cameras, calibration is a critical stage and must be completed before capturing data (Baker and Aloimonos, 2003). Therefore, various camera calibration methods have been published, suited to different project purposes.

This chapter aims to provide a literature review of the previous camera calibration methods and discuss both the benefits and drawbacks of these methods. These related works are considered and analysed thoroughly to determine the most suitable approach for our project purpose.

2.1 Camera calibration

The actual camera calibration process is performed using calibration targets. These targets have a definite structure or sample, and their geometry is known for great precision. The camera will take the image of these calibration targets and then calculate the camera parameters (intrinsic and extrinsic) using math formulas and algorithms. One of the most notable publications on camera calibration is by Tsai (1987); he presented a two-stage camera calibration method using a mono view co-planar set of points. Zhang (2000) proposed a calibration method using Closed-Form solution and Maximum-Likelihood Estimation. In this approach, his calibration

target was a planar board with mostly square patterns and circle patterns at the corners, printed on paper and stuck onto the board. Research by Triggs (1999) showed that the camera is required to capture at least five 3D points to one image to calibrate the extrinsic parameters. There is a drawback in Triggs' approach that these five 3D calibration dots had to belong to at least two different planes.

2.2 Multi-camera calibration

2.2.1 Stereo cameras

Unlike calibrating a single camera, the calibration technique for a system with several cameras is more complicated and troublesome. In this situation, the intrinsic parameters must be calibrated independently, while the extrinsic calibration process should be implemented simultaneously. Various calibration methods have been published to estimate the extrinsic parameters of a multiple camera system. In research in 2003, Tomita and Ueshiba introduced a calibration method for two cameras using a planar calibration target. This pair of cameras would capture images of the target from two different orientations. Then, applying homography, both cameras would be intrinsically and extrinsically calibrated, and all the camera parameters would be solved (Ueshiba and Tomita, 2003). With this method, although capturing images from two different orientations is enough, we can improve the accuracy of the calibration process by capturing more images from various locations. This method was then developed by calibrating a system with twelve cameras precisely.

2.2.2 Non-overlapping Field of View cameras

However, the calibration method of Tomita & Ueshiba contained a significant drawback. This approach only works with a multiple overlapping camera system, which means all the cameras in the system have to be able to capture the 2D calibration target fully. In a research in 2016, Dong et al. (2016) introduced a method to calibrate a non-overlapping camera system. He used a wide planar board and printed encoded patterns with predefined positions (relative to the board origin). There-

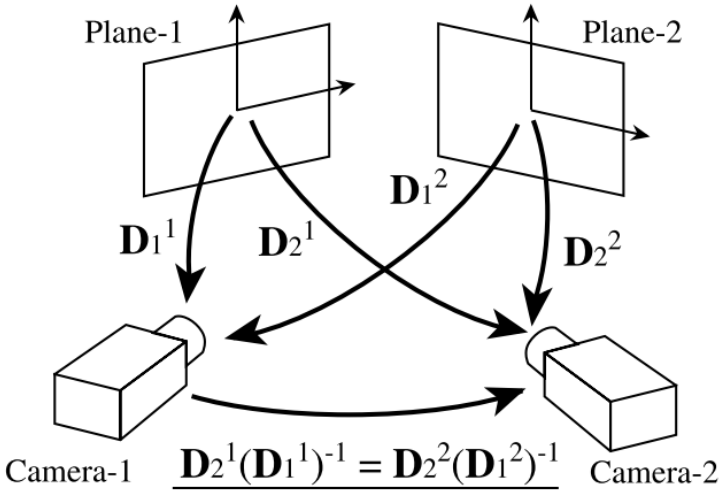

Figure 2.1: Ueshiba & Tomita (2003) calibration method for overlapping camera system

fore, each camera in the system does not need to capture the whole board during calibration, and camera can identify every single encoded pattern using a binary color-coding system. However, this method can only apply to extrinsically calibrate the camera system, so Dong had to get the intrinsic parameters of his cameras using Zhang's method.

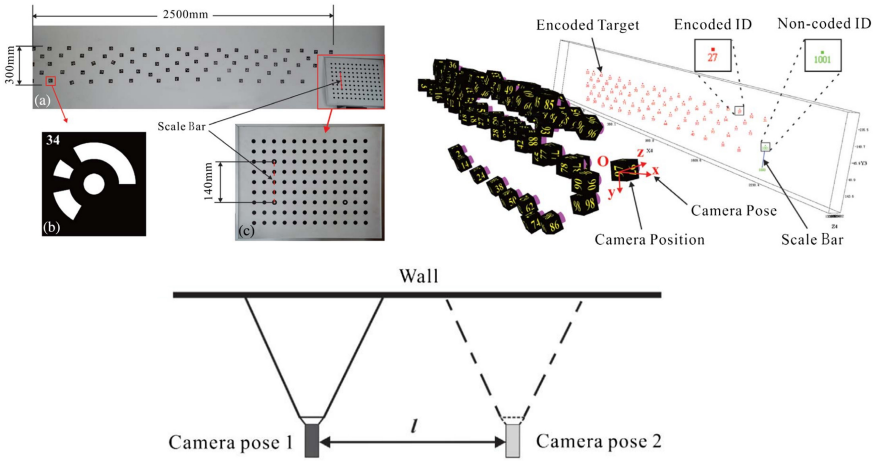

Figure 2.2: Dong et al. (2016) method for non-overlapping multiple camera system

Dong's calibration method still requires the cameras are placed close together with the same orientation due to the particular characteristics of the 2D calibration target. In fact, depending on the project purpose, there will be cases where the

system contains cameras facing each other, and 2D calibration targets cannot be used because all cameras cannot thoroughly capture the planar calibration target. Marcon et al. (2017) introduced a calibration method that used a double-sided checkerboard. The cameras having opposite orientations independently capture one side of the board, and then the extrinsic parameters are estimated together with the displacement error between two sides of the board.

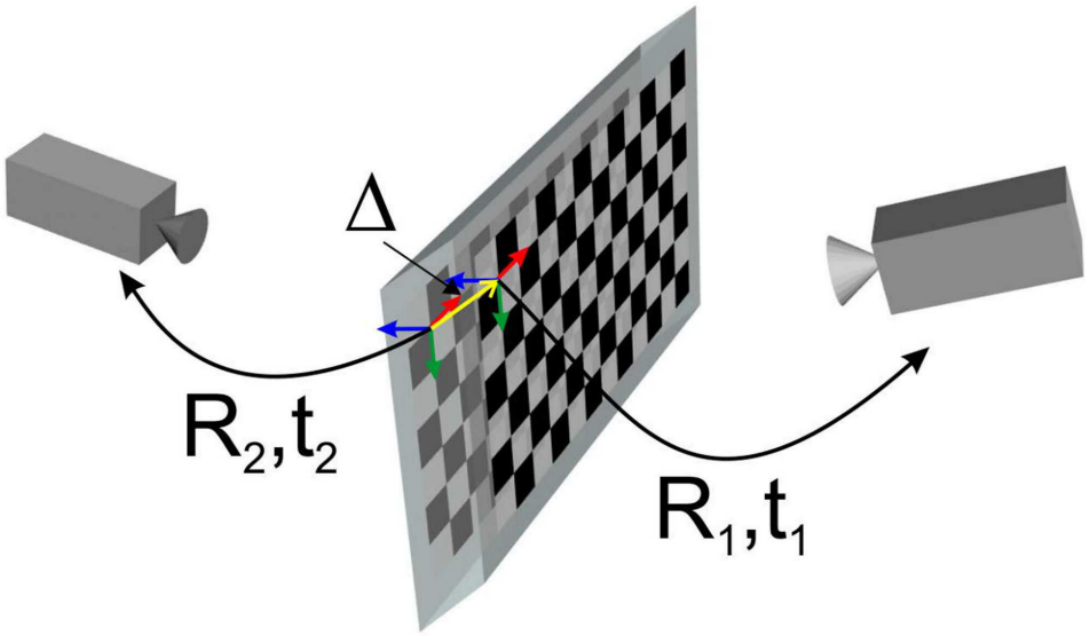

Figure 2.3: Double-sided checkerboard of Marcon et al. (2017)

However, Marcon's (2017) approach could not solve the problem of cameras facing back to each other, for example, the airborne remote sensing camera system. Therefore, Yin et al. (2018) proposed a calibration method as a solution for these multi-camera systems. In his approach, he chose one camera as the origin reference frame, and then the extrinsic parameters would be the relative pose between each camera in the system to the reference camera. His method consisted of two stages: estimating the relative poses between every pair of cameras in this system and then calculating the extrinsic parameters based on optimising the re-projection errors using the Global Optimal Method. The results obtained were very positive: 0.001 radians and 0.08 millimetres on rotation and translation, respectively.

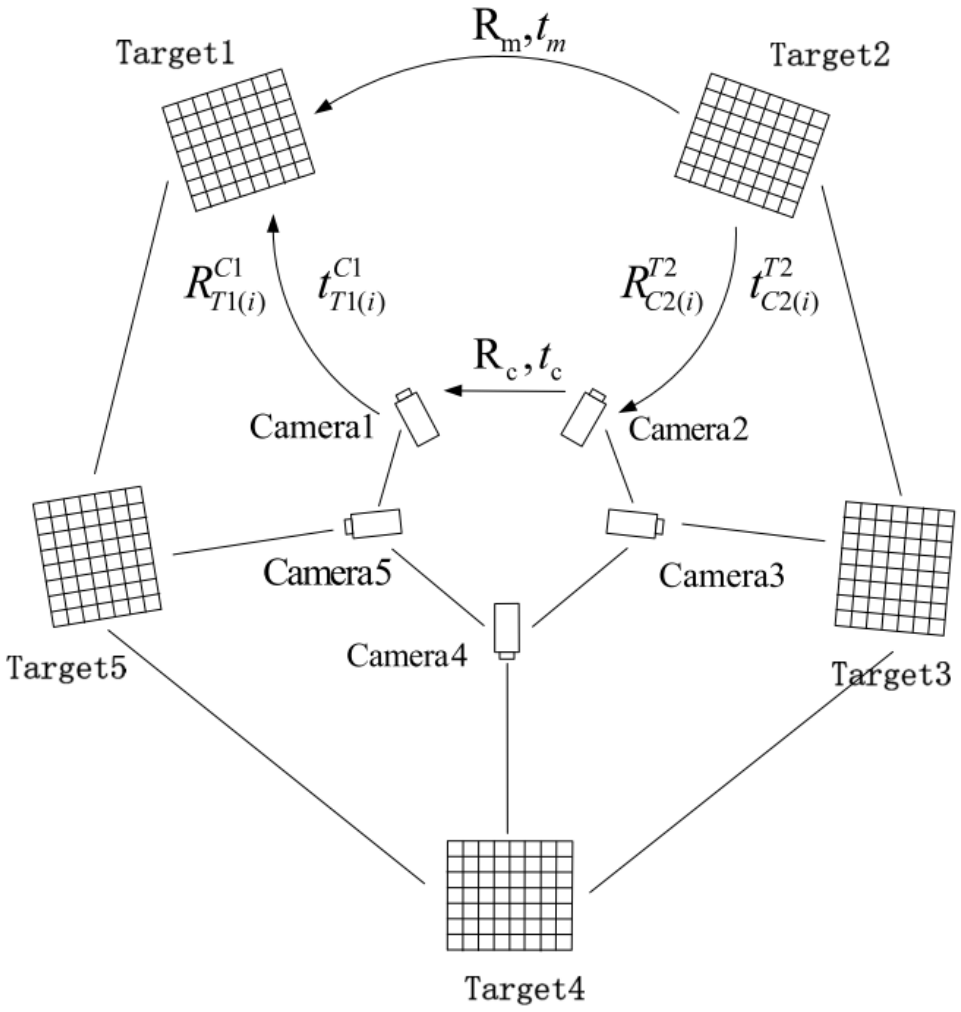

Figure 2.4: Yin et al. (2018) calibration method

Additionally, several calibration methods use non-planar calibration targets or 3D calibration objects. Shen and Hornsey (2011) published a technique using a sphere to calibrate a multi-camera system. They used eight 3D calibration dots, which are spherical with different colors, for the calibration of a system with twelve cameras. In this approach, each camera does not need to capture all eight dots in the calibration process, because just six calibration points are required to be captured by each camera once.

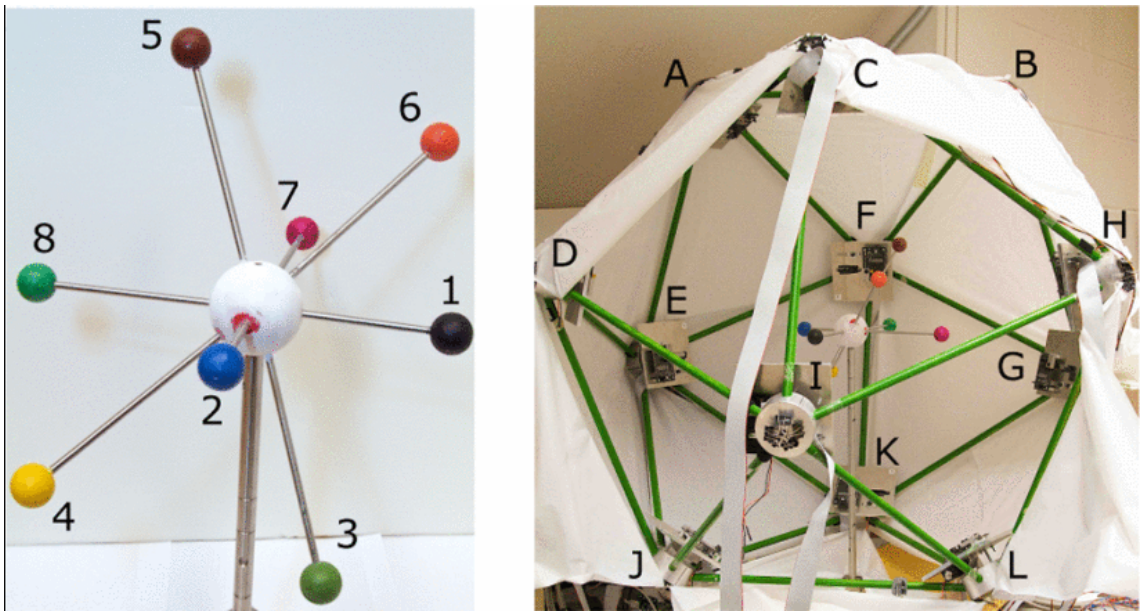

Figure 2.5: Spherical calibration targets of Shen & Hornsey (2011)

2.3 Fiducial markers

Fiducial markers play an essential role in camera calibration. Fiducial marker defines visual sample targets within a physical environment that is discoverable and localisable by kinds of an exteroceptive sensor (camera, lidar...) (Korthals et al., 2019). Numerous types of fiducial markers were proposed in computer vision systems, such as ArUco markers (Garrido-Jurado et al., 2014), AprilTag (Olson, 2011), ARToolKit... Fiducial markers have been widely used in various vision applications, including camera calibration, robotics and augmented reality. Moreover, many support libraries have been provided to simplify the adoption of these markers.

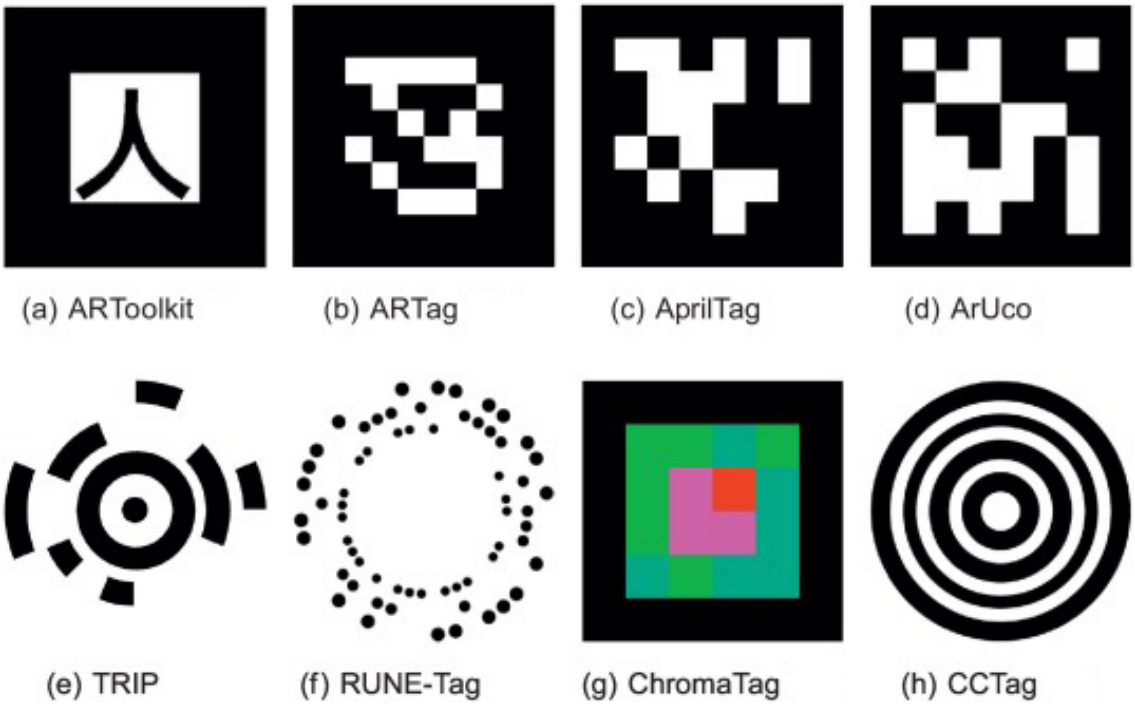

Figure 2.6: Fiducial markers

2.4 General Graph Optimisation

To increase the accuracy of camera parameters during the calibration process, especially calibrating a multi-camera system, we should repeat the calibration procedure many times to optimise the result. The problem of graph optimisation has been researched extensively in several computer vision and robotics fields. Reinke et al. (2019) have proposed an extrinsic calibration method for a multi-camera system on legged robots based on factor graph optimisation. In their experiments, they captured a total of 1312 images, and after calibrating, their results had errors of about 2.03% and 3.8% on translation and rotation, respectively.

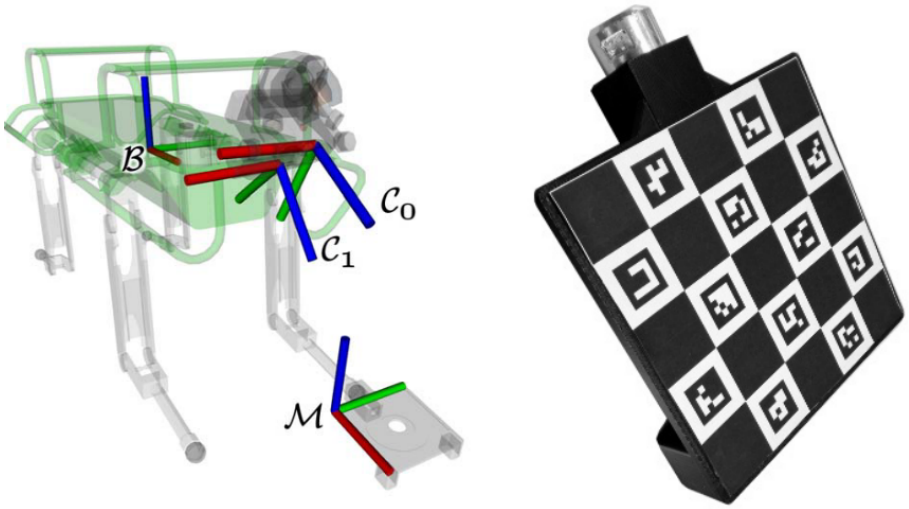

Figure 2.7: Multiple camera system on a legged robot

Chapter 3

Approach

This capstone report aims to develop a method to calibrate the extrinsic parameters of a multiple partially overlapping RGB-D camera system. Each camera in the system already has the intrinsic parameters (from the manufacturer), so this report just focuses on extrinsically calibrating. The approach consists of two main parts: the first section proposes the double-sided ArUco board as the calibration target and estimates the relative pose between the board and each camera in the system, and the second part mentions the application of the General Graph Optimisation in our method.

3.1 Camera model

It is essential to obtain a good established mathematical modelling that correctly defines the operating of a typical camera to handle the calibration. In this method, the pinhole camera model is used to describe a 3D point $P(X, Y, Z)$ in the global frame to a 2D image plane position $p(u, v)$ via a projection matrix M.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The projection matrix M combines the intrinsic parameters K and the extrinsic parameters $[R, T]$ for a camera in a specific setup.

$$M = K[RT]$$

To determine the relative pose between cameras in the system (also known as the extrinsic parameters), firstly, we have to estimate the transformation from the calibration target (in this case is the ArUco board) to each camera. To optimise the calibration result, this work is performed continuously throughout the calibration process. All the transformation (translation and rotation) between the cameras to the ArUco board during the calibration process are recorded for the Pose Graph Optimisation.

The Perspective-n-Point (PnP) pose computation problem is reviewed to detect the ArUco board pose respected to the camera frame. This method estimates the relative transform (translation and rotation) between a calibrated camera (with all the intrinsic parameters) and a 3D target, minimising the re-projection error from the 2D-3D point.

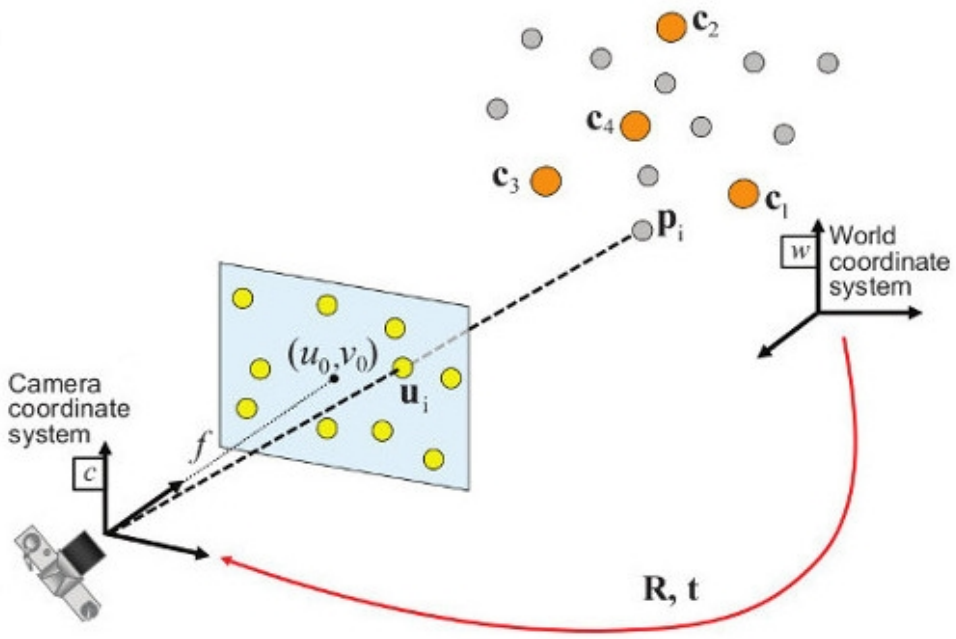

Figure 3.1: Perspective-n-Point (PnP) pose problem

Theoretically, we have a set of 3D points within a global frame of reference and their respective 2D image projections. Assuming the camera has been intrinsically calibrated already and the camera matrix is defined, we are able to determine the relative pose (position and orientation) between the camera itself and the global frame. This follows the projecting model in perspective for the cameras:

$$p_I = K[RT]p_W$$

In which:

$p_W = [X_W, Y_W, Z_W, 1]$: the global point coordinates in homogeneous form.

$p_I = [u, v, 1]$: the respective image point coordinates in homogeneous form.

$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ a matrix includes the intrinsic parameters, also known as camera matrix. f_x, f_y is the focal length and c_x, c_y is the principal point.

R, T: 3D rotation and translation matrix (extrinsic parameters).

This results in the formula equation of the model:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

Therefore, the relative pose combines the 3D translation and the 3D rotation, which transforms a 3D object in the global frame into the camera frame:

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

3.2 Calibration target configuration: double-sided ArUco board

We propose a double-sided planar board configuration with a set of ArUco markers as the calibration target. The board's thickness is carefully measured to the nearest 0.1 millimetres. The ArUco markers are printed on paper and mounted on both sides of the board, following the given order. These markers are the same size with given side lengths. In addition, to simplify the detection process, the $[x, y, z]$ coordinates of 4 corners of the ArUco markers are determined, which is respected to the board's origin frame. Finally, the origin coordinate frame of the double-sided board is identified in the lower-left corner of the front side of the board.

Figure 3.2: ArUco double-sided board configuration

3.3 Pose Graph Optimisation

We propose creating a pose graph, as shown in Figure 3.3. The pose graph consists of nodes that are connected by edges, which determines the relative pose between any two corresponding nodes in the graph and can contain uncertainty information of the measurement. The white nodes $X_k, k \in \{1, 2, ..., M\}$ denote the cameras in our system, and the grey nodes $L_i, i \in \{1, 2, ..., N\}$ represent the double-sided ArUco board's origin poses that are detected by at least two cameras. Note that the board pose does not need to be detected by all cameras. Therefore, in the graph each board pose node (grey) must be connected with at least two camera nodes (white), with the edges being the relative poses between the board's origin and the corresponding camera.

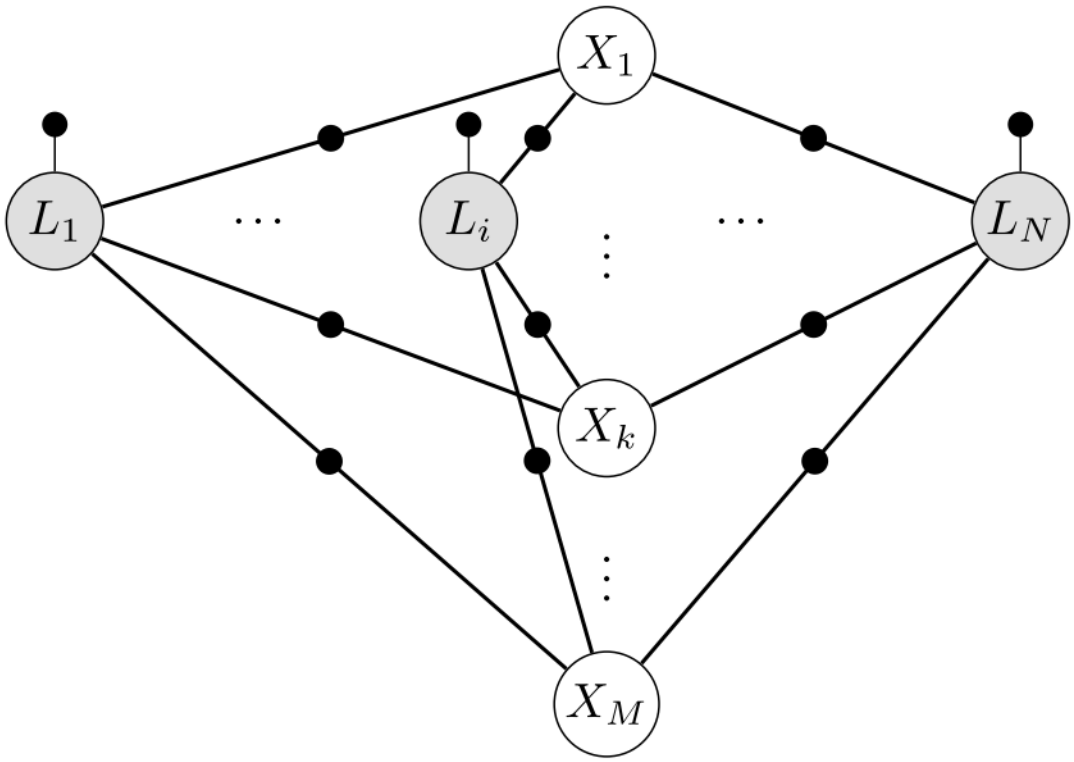

Figure 3.3: Pose Graph represents the calibration method

Chapter 4

Evaluation and Results

This chapter describes the experimental process of this camera calibration method and the equipment used for the experiments. Furthermore, all the system parameters and results are also demonstrated in detail. Even though the system contains 16 cameras, we worked with four cameras in this experiment scope to test the calibration method. If the experiment result is accurate, we will gradually increase the number of cameras for further experiments.

4.1 Overview

Firstly, this chapter mentions the equipment necessary for the experiment, especially the cameras and the double-sided ArUco board. Cameras with their intrinsic parameters, as well as the ArUco board configuration, are defined clearly.

This chapter then demonstrates the extrinsic calibration process for each camera. In order to optimise the calibration result, each camera continuously captures the image of the ArUco board and estimates the board's transformation respecting to the camera frame during the calibration process. Finally, the set of estimated board poses of the whole camera system from the previous stage is processed by the General Graph Optimisation (G2O) framework. This framework aggregates the detected board's poses of the entire procedure and then calculates the final extrinsic parameters between cameras in the system.

This experiment's program framework and algorithm are mainly programmed in C++ with Open-CV 4.2.0 library. MATLAB programming language is also used to test the algorithm and initialise the ArUco board configuration. Besides, we use ROS Noetic as a middle-ware to help communicate between the camera system and our framework.

4.2 Camera system

For the scope and purpose of this experiment, we decide to use four RGB-D cameras Intel Realsense RGB-D D435i. This type of camera can capture images up to 1920x1080 resolution with a frame rate up to 60 Hz. The RGB-D camera can also capture depth images, which is very important for our project purpose, such as 3D reconstruction. Furthermore, the manufacturers have calibrated the intrinsic parameters of these cameras, so we just have to focus on the extrinsic calibration problem. Additionally, these cameras can be synchronised together via cable, which is important in working with multiple cameras. Intel has developed a ROS "wrapper" for the camera firmware that makes it more straightforward to communicate between the cameras and our framework, simplifying the programming for the calibration algorithm.

Figure 4.1: RGB-D Camera Intel Realsense D435i

The intrinsic parameter (also known as the camera matrix) is defined by

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

in which f_x, f_y is the focal length and (c_x, c_y) is the principal point, which is unique to each camera. After checking the manufacturer firmware, the intrinsic parameters of the four cameras in our system are listed below. We intend to store the intrinsic parameters of these cameras in a YAML file to simplify our program framework as well as be convenient for modifying the parameters without recompiling the code.

Table 4.1: Intrinsic parameters of each camera.

Cam	f_x	f_y	c_x	c_y
1	633.470275878906	633.470275878906	643.229064941406	358.437103271484
2	637.565185546875	637.565185546875	643.337707519531	355.360290527343
3	640.793884277343	640.793884277343	645.108825683593	356.419738769531
4	642.689758300781	642.689758300781	645.234985351562	355.756347656250

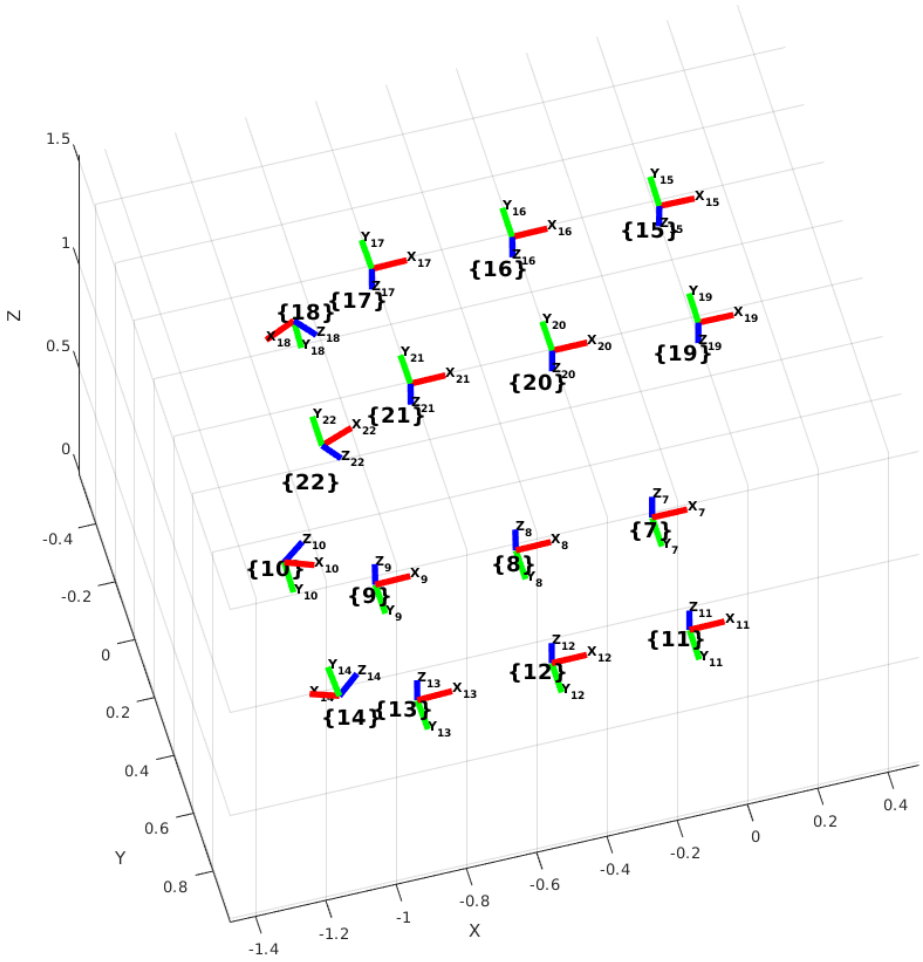

Figure 4.2: Camera layout

4.3 Double-sided ArUco board configuration

In this experiment, firstly, we use a wood panel with an exact thickness of 6.6 millimetres (measured carefully). The board should be large enough to fit an A3 paper containing a set of ArUco markers, as well as make it easier for cameras to detect the board's features.

Secondly, we decided to use 56 ArUco markers with IDs from 0 to 55, which means 28 tags on each side of the board. The markers' size is 4x4, composed of 16 binary bits. All 56 markers are printed on two sheets of A3 size paper, with 28 markers each, ordered from left to right and top to bottom with four rows and seven columns. Each marker is a square with a four-centimetre side length, and the gap between each marker is two centimetres. We use Microsoft Publisher to design the ArUco board before printing to ensure the size of each marker, as well as the distance between the markers, is correct. Additionally, when sticking on the wood board, we must guarantee that the markers on the two sides match together precisely.

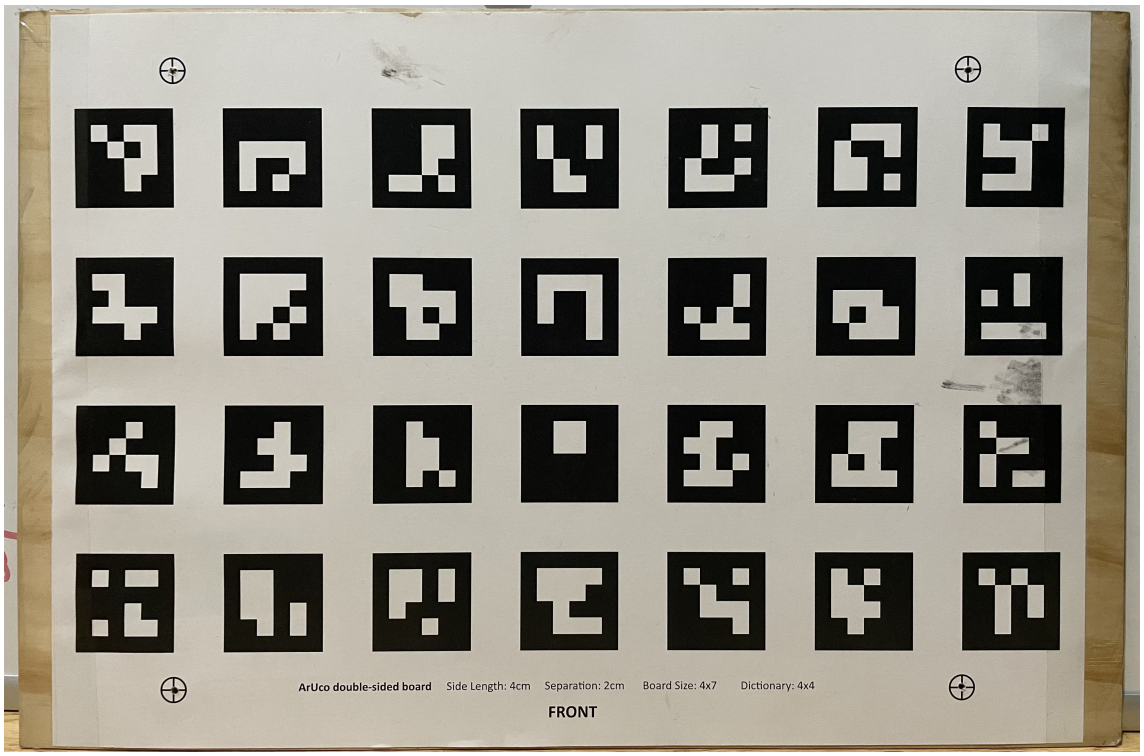

Figure 4.3: Double-sided ArUco board (front side)

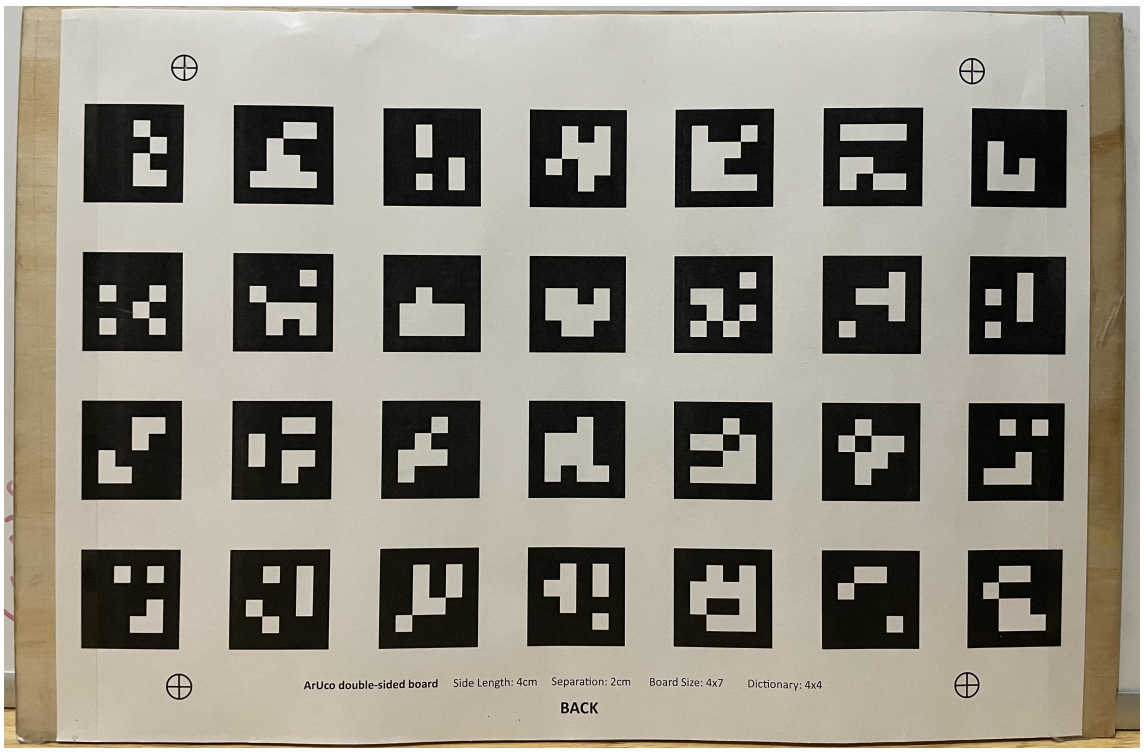

Figure 4.4: Double-sided ArUco board (back side)

Finally, we have to initialise the parameters of the ArUco board configuration. As mentioned above, a board's layout includes three essential parameters: IDs of markers, the object points and the dictionary. IDs are numbered from 0 to 55, and each ID represents a marker on the board. Because the ArUco markers we are using have a 4x4 size and the total of tags used is 56, we decided to use the dictionary DICT_4X4_100 (each dictionary is contained a different amount of ArUco tags). About the object points, we use a MATLAB script to generate a list of object points belonging to the markers (the table below). Thanks to Mr Jasprabhjit Mehami for the support on this Matlab script.

The origin coordinate of the whole board is located at the bottom-left corner of ArUco marker ID number 21. Board's thickness is 6.6 millimetres (0.0066 meters), so all the ArUco markers located on the backside have the z-coordinate of corners equal to -0.0066 (which is respected to the board origin coordinate).

Table 4.2: Double-sided ArUco board configuration

IDs	x	y	z	x	y	z	x	y	z	x	y	z
0	0	0.22	0	0.04	0.22	0	0.04	0.18	0	0	0.18	0
1	0.06	0.22	0	0.1	0.22	0	0.1	0.18	0	0.06	0.18	0
2	0.12	0.22	0	0.16	0.22	0	0.16	0.18	0	0.12	0.18	0
3	0.18	0.22	0	0.22	0.22	0	0.22	0.18	0	0.18	0.18	0
4	0.24	0.22	0	0.28	0.22	0	0.28	0.18	0	0.24	0.18	0
5	0.3	0.22	0	0.34	0.22	0	0.34	0.18	0	0.3	0.18	0
6	0.36	0.22	0	0.4	0.22	0	0.4	0.18	0	0.36	0.18	0
7	0	0.16	0	0.04	0.16	0	0.04	0.12	0	0	0.12	0
8	0.06	0.16	0	0.1	0.16	0	0.1	0.12	0	0.06	0.12	0
9	0.12	0.16	0	0.16	0.16	0	0.16	0.12	0	0.12	0.12	0
10	0.18	0.16	0	0.22	0.16	0	0.22	0.12	0	0.18	0.12	0
11	0.24	0.16	0	0.28	0.16	0	0.28	0.12	0	0.24	0.12	0
12	0.3	0.16	0	0.34	0.16	0	0.34	0.12	0	0.3	0.12	0
13	0.36	0.16	0	0.4	0.16	0	0.4	0.12	0	0.36	0.12	0
14	0	0.1	0	0.04	0.1	0	0.04	0.06	0	0	0.06	0
15	0.06	0.1	0	0.1	0.1	0	0.1	0.06	0	0.06	0.06	0
16	0.12	0.1	0	0.16	0.1	0	0.16	0.06	0	0.12	0.06	0
17	0.18	0.1	0	0.22	0.1	0	0.22	0.06	0	0.18	0.06	0
18	0.24	0.1	0	0.28	0.1	0	0.28	0.06	0	0.24	0.06	0
19	0.3	0.1	0	0.34	0.1	0	0.34	0.06	0	0.3	0.06	0
20	0.36	0.1	0	0.4	0.1	0	0.4	0.06	0	0.36	0.06	0
21	0	0.04	0	0.04	0.04	0	0.04	0	0	0	0	0
22	0.06	0.04	0	0.1	0.04	0	0.1	0	0	0.06	0	0
23	0.12	0.04	0	0.16	0.04	0	0.16	0	0	0.12	0	0
24	0.18	0.04	0	0.22	0.04	0	0.22	0	0	0.18	0	0
25	0.24	0.04	0	0.28	0.04	0	0.28	0	0	0.24	0	0
26	0.3	0.04	0	0.34	0.04	0	0.34	0	0	0.3	0	0
27	0.36	0.04	0	0.4	0.04	0	0.4	0	0	0.36	0	0
28	0.36	0.22	-0.0066	0.4	0.22	-0.0066	0.4	0.18	-0.0066	0.36	0.18	-0.0066
29	0.3	0.22	-0.0066	0.34	0.22	-0.0066	0.34	0.18	-0.0066	0.3	0.18	-0.0066
30	0.24	0.22	-0.0066	0.28	0.22	-0.0066	0.28	0.18	-0.0066	0.24	0.18	-0.0066
31	0.18	0.22	-0.0066	0.22	0.22	-0.0066	0.22	0.18	-0.0066	0.18	0.18	-0.0066
32	0.12	0.22	-0.0066	0.16	0.22	-0.0066	0.16	0.18	-0.0066	0.12	0.18	-0.0066

33	0.06	0.22	-0.0066	0.1	0.22	-0.0066	0.1	0.18	-0.0066	0.06	0.18	-0.0066
34	0	0.22	-0.0066	0.04	0.22	-0.0066	0.04	0.18	-0.0066	0	0.18	-0.0066
35	0.36	0.16	-0.0066	0.4	0.16	-0.0066	0.4	0.12	-0.0066	0.36	0.12	-0.0066
36	0.3	0.16	-0.0066	0.34	0.16	-0.0066	0.34	0.12	-0.0066	0.3	0.12	-0.0066
37	0.24	0.16	-0.0066	0.28	0.16	-0.0066	0.28	0.12	-0.0066	0.24	0.12	-0.0066
38	0.18	0.16	-0.0066	0.22	0.16	-0.0066	0.22	0.12	-0.0066	0.18	0.12	-0.0066
39	0.12	0.16	-0.0066	0.16	0.16	-0.0066	0.16	0.12	-0.0066	0.12	0.12	-0.0066
40	0.06	0.16	-0.0066	0.1	0.16	-0.0066	0.1	0.12	-0.0066	0.06	0.12	-0.0066
41	0	0.16	-0.0066	0.04	0.16	-0.0066	0.04	0.12	-0.0066	0	0.12	-0.0066
42	0.36	0.1	-0.0066	0.4	0.1	-0.0066	0.4	0.06	-0.0066	0.36	0.06	-0.0066
43	0.3	0.1	-0.0066	0.34	0.1	-0.0066	0.34	0.06	-0.0066	0.3	0.06	-0.0066
44	0.24	0.1	-0.0066	0.28	0.1	-0.0066	0.28	0.06	-0.0066	0.24	0.06	-0.0066
45	0.18	0.1	-0.0066	0.22	0.1	-0.0066	0.22	0.06	-0.0066	0.18	0.06	-0.0066
46	0.12	0.1	-0.0066	0.16	0.1	-0.0066	0.16	0.06	-0.0066	0.12	0.06	-0.0066
47	0.06	0.1	-0.0066	0.1	0.1	-0.0066	0.1	0.06	-0.0066	0.06	0.06	-0.0066
48	0	0.1	-0.0066	0.04	0.1	-0.0066	0.04	0.06	-0.0066	0	0.06	-0.0066
49	0.36	0.04	-0.0066	0.4	0.04	-0.0066	0.4	0	-0.0066	0.36	0	-0.0066
50	0.3	0.04	-0.0066	0.34	0.04	-0.0066	0.34	0	-0.0066	0.3	0	-0.0066
51	0.24	0.04	-0.0066	0.28	0.04	-0.0066	0.28	0	-0.0066	0.24	0	-0.0066
52	0.18	0.04	-0.0066	0.22	0.04	-0.0066	0.22	0	-0.0066	0.18	0	-0.0066
53	0.12	0.04	-0.0066	0.16	0.04	-0.0066	0.16	0	-0.0066	0.12	0	-0.0066
54	0.06	0.04	-0.0066	0.1	0.04	-0.0066	0.1	0	-0.0066	0.06	0	-0.0066
55	0	0.04	-0.0066	0.04	0.04	-0.0066	0.04	0	-0.0066	0	0	-0.0066

4.4 Board pose estimation

To estimate the relative pose of the ArUco board to each camera, we have to solve the Perspective-n-Point (PnP) pose problem. Due to the popularity of this problem in camera calibration, several methods are implemented and published as packages or libraries. Our program framework is mainly written in C++, so we intend to use the Open-CV library for Perspective-n-Point (PnP) pose computation.

Figure 4.5: SolvePnP() function in Open-CV library in detail

In which, "objectPoints" are corner coordinates of ArUco markers respected to the board's origin, and the "imagePoints" are the corresponding corner points in the image. Besides, "cameraMatrix" is the matrix that contains intrinsic parameters of the calibrated camera, mentioned in the above chapter (4.1). On the other hand, "rvec" and "tvec" are the rotation and translation vectors, respectively, demonstrating the relative pose between the ArUco board and camera.

Firstly, we must detect the ArUco markers' corner position of the image (in pixel coordinate). Technically, we do not need to catch all the markers to estimate the board's pose. Detected markers are identified along with the given IDs, which are defined by the ArUco dictionary.

After detecting markers and their IDs on the image, we have to determine their corresponding 3D points respected to ArUco board pose. As mentioned above (chapter 4.2), we already have had a list of markers' corner 3D coordinates (the object points) initialising by MATLAB. Based on the list of markers detected on the image and their IDs, we can define their relative 3D point coordinates in the

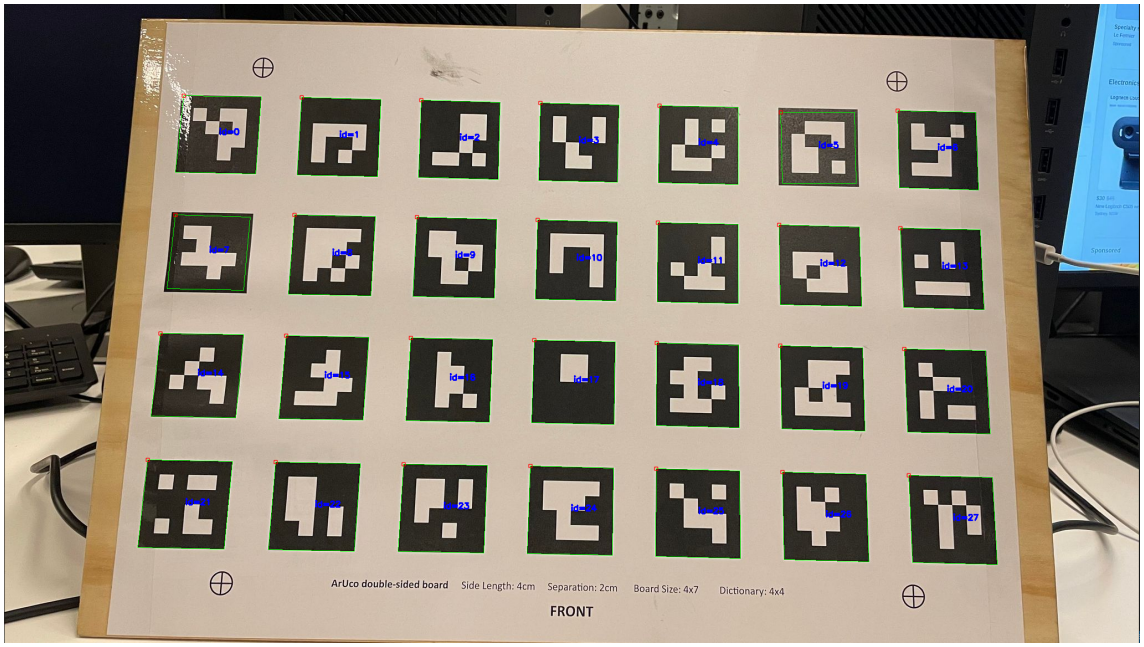

Figure 4.6: ArUco markers detection in image

board's origin frame (see Table 4.2). Finally, after detecting the image points and the object points of ArUco markers, we should combine them with the intrinsic parameters of the respective camera (see Table 4.1). Using the solvePnP() function, we can calculate the 3D rotation and translation vector from the ArUco board pose to the respective camera, and then estimate the relative pose between them. The demonstration of the ArUco board estimation process is shown below.

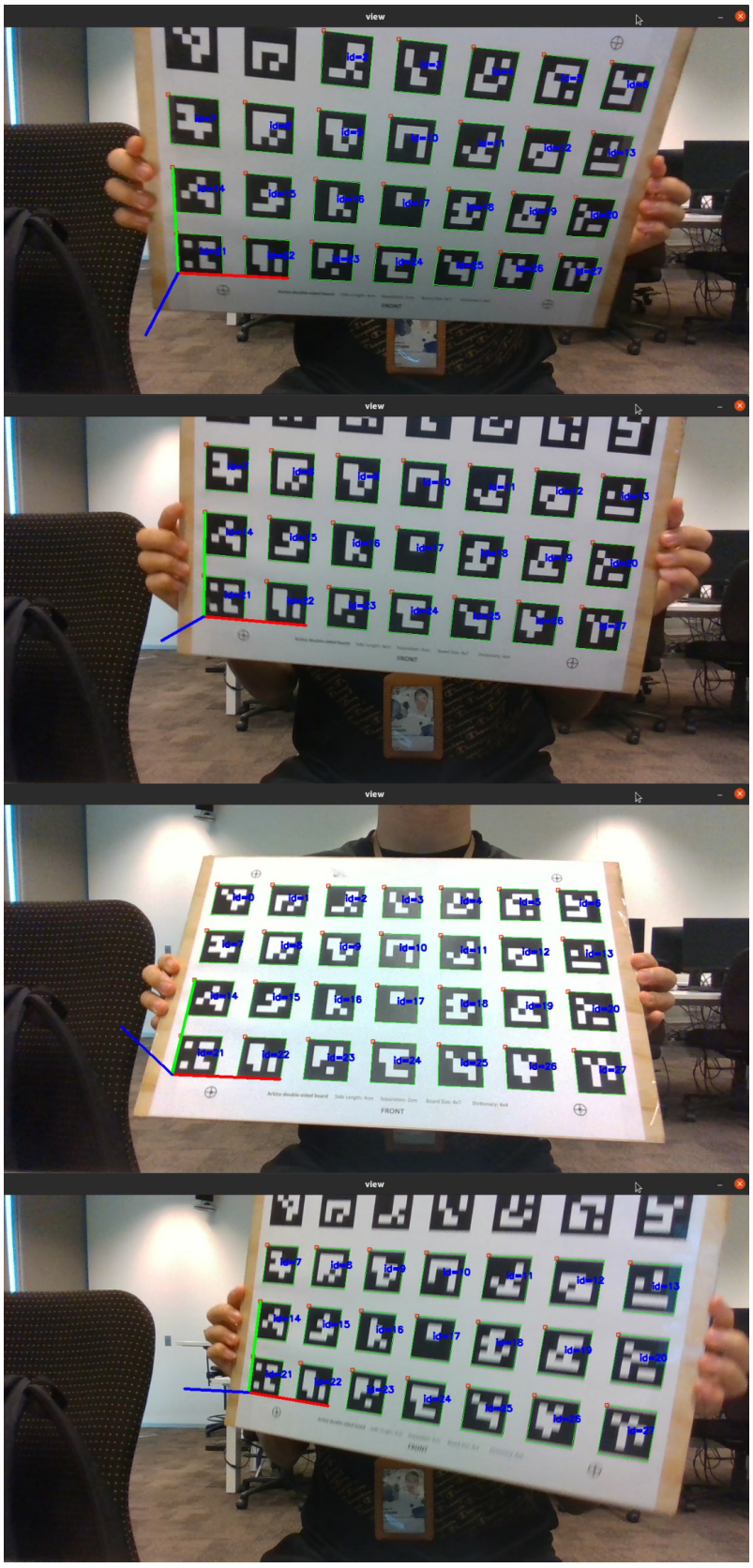

Figure 4.7: ArUco board estimation (front side)

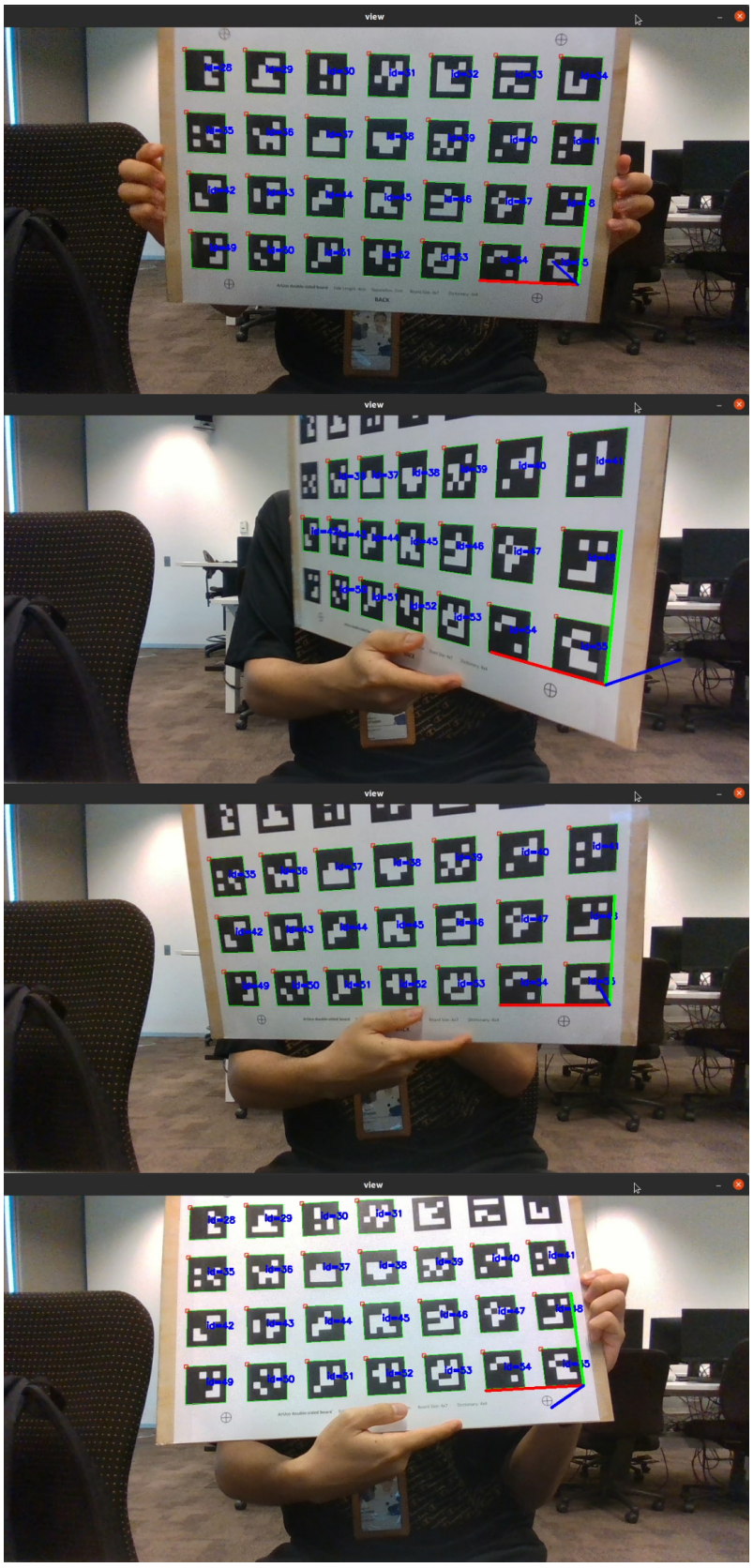

Figure 4.8: ArUco board estimation (back side)

4.5 Pose Graph Optimisation

After estimating the relative pose between the double-sided ArUco board and each camera in the system, we separated the transformation into translation and rotation. The translation is presented as 3D vectors $[t_x, t_y, t_z]$, and the rotation is demonstrated as the quaternion formula $[q_x, q_y, q_z, q_w]$. We captured a total of 1652 images during the calibration process, which means each camera captured 413 photos of the ArUco board.

To implement the pose graph optimisation, we saved the estimated translation vectors and rotation quaternions of the whole calibration process as two CSV files: Nodes.csv and Edges.csv.

One CSV file contained the nodes of the pose graph, including four nodes of cameras and 413 nodes of image poses. Camera 7 is the origin node, with the default value $[t_x, t_y, t_z] = [0, 0, 0]$ and $[q_x, q_y, q_z, q_w] = [0, 0, 0, 1]$. The remaining three camera nodes are the initial guess relative poses, which are measured directly from the multi-camera system. Finally, the image pose nodes are the transformations of the board's pose to the origin node, camera 7, which are estimated in the previous stage.

The other CSV file included the edges of the graph, which are the relative between nodes in the graph. We presented the edges between camera nodes and image pose nodes, which are the extrinsic parameters of each camera estimated from the ArUco board. In case the camera in our system could not detect the board, we used the last known pose of the previous detectable image. The edge values are also saved as 3D vectors $[t_x, t_y, t_z]$, and the rotation is demonstrated as the quaternion formula $[q_x, q_y, q_z, q_w]$.

Node ID	tx	ty	tz	qx	qy	qz	qw
7	0	0	0	0	0	0	1
8	-0.2	0	0	0	0	0	1
9	-0.4	0	0	0	0	0	1
10	-0.6	0	0.1	0	0.4	0	0.9
23	-0.334	0.239252	0.829421	-0.03693	0.008331	0.997229	-0.06404
24	-0.33316	0.23901	0.830862	-0.03655	0.00867	0.997221	-0.06433
25	-0.33206	0.238901	0.831768	-0.03589	0.007654	0.997265	-0.06416
26	-0.33158	0.238777	0.832079	-0.03716	0.009308	0.997186	-0.06445
27	-0.33286	0.238183	0.832489	-0.03588	0.010248	0.997226	-0.06441
28	-0.33654	0.237243	0.835552	-0.03676	0.009733	0.997109	-0.06579
29	-0.33937	0.235297	0.834574	-0.0374	0.010044	0.997047	-0.06631
30	-0.34534	0.23413	0.837399	-0.0439	0.009288	0.996712	-0.06746
31	-0.35184	0.232812	0.839414	-0.04857	0.008404	0.996448	-0.06828
32	-0.35974	0.231641	0.843649	-0.05504	0.009756	0.995995	-0.06979
33	-0.36707	0.229844	0.84621	-0.05518	0.007652	0.995826	-0.07229
34	-0.37648	0.228179	0.84903	-0.05633	0.010761	0.99556	-0.07465
35	-0.38687	0.226576	0.851375	-0.05671	0.010582	0.995373	-0.07684
36	-0.39697	0.22458	0.852005	-0.05858	0.015627	0.995305	-0.07545
37	-0.40699	0.223257	0.850928	-0.05758	0.014714	0.99535	-0.0758

Figure 4.9: Nodes CSV file

In the Nodes.csv file, the rows list the nodes present in the pose graph and the transformation between the first node (node 7) and the corresponding node in this row. The columns describe the transformation components, including 3D translation vectors $[t_x, t_y, t_z]$ and rotation quaternion $[q_x, q_y, q_z, q_w]$.

Cam	Node ID	tx	ty	tz	qx	qy	qz	qw
7	23	-0.334	0.239252	0.829421	-0.03693	0.008331	0.997229	-0.06404
7	24	-0.33316	0.23901	0.830862	-0.03655	0.00867	0.997221	-0.06433
7	25	-0.33206	0.238901	0.831768	-0.03589	0.007654	0.997265	-0.06416
7	26	-0.33158	0.238777	0.832079	-0.03716	0.009308	0.997186	-0.06445
7	27	-0.33286	0.238183	0.832489	-0.03588	0.010248	0.997226	-0.06441
7	28	-0.33654	0.237243	0.835552	-0.03676	0.009733	0.997109	-0.06579
7	29	-0.33937	0.235297	0.834574	-0.0374	0.010044	0.997047	-0.06631
7	30	-0.34534	0.23413	0.837399	-0.0439	0.009288	0.996712	-0.06746
8	23	0.067101	0.239014	0.832156	-0.03917	0.007246	0.997285	-0.06193
8	24	0.068314	0.238624	0.833288	-0.03781	0.005123	0.997353	-0.0619
8	25	0.069519	0.238447	0.833424	-0.03742	0.006478	0.997382	-0.06153
8	26	0.069445	0.238286	0.832715	-0.03737	0.010057	0.997356	-0.0615
8	27	0.067672	0.237394	0.833961	-0.03733	0.009788	0.997321	-0.06214
8	28	0.064689	0.236097	0.834696	-0.03518	0.006537	0.997354	-0.06328
8	29	0.060229	0.234697	0.836247	-0.04132	0.010296	0.997019	-0.06434
8	30	0.054345	0.233299	0.838123	-0.0454	0.006351	0.996809	-0.06535
9	23	0.469591	0.238205	0.830475	-0.03499	0.008408	0.997604	-0.05909
9	24	0.470781	0.23791	0.831675	-0.03429	0.007964	0.997648	-0.05882
9	25	0.471774	0.237843	0.831858	-0.03303	0.008093	0.997688	-0.05884
9	26	0.471515	0.237417	0.832411	-0.03503	0.00847	0.997628	-0.05864
9	27	0.469753	0.236485	0.8327	-0.03297	0.008223	0.99768	-0.05899
9	28	0.466891	0.235558	0.834235	-0.03217	0.008271	0.997594	-0.06086
9	29	0.46233	0.234039	0.835493	-0.03732	0.009153	0.997362	-0.06159
9	30	0.456502	0.232462	0.83684	-0.04284	0.010055	0.997103	-0.06204
10	23	0.284251	0.213917	0.942096	-0.30528	0.007922	0.950483	-0.05765
10	24	0.284772	0.213489	0.943017	-0.30375	0.007476	0.950982	-0.05759
10	25	0.285725	0.213507	0.944912	-0.30253	0.006792	0.951362	-0.05778
10	26	0.284902	0.212997	0.944319	-0.30446	0.007435	0.95076	-0.05752
10	27	0.283151	0.212401	0.944008	-0.30383	0.009146	0.95091	-0.05807
10	28	0.28066	0.210735	0.944433	-0.30311	0.007554	0.951124	-0.05855
10	29	0.276064	0.209643	0.941833	-0.30661	0.009683	0.949881	-0.06019
10	30	0.2704	0.207987	0.9405	-0.31182	0.010068	0.948182	-0.06013
10	31	0.262956	0.206717	0.938527	-0.31722	0.009744	0.946337	-0.06103

Figure 4.10: Edges CSV file

In the Edges.csv file, the first two columns represent two nodes of each edge, and the remaining columns are the relative poses between these two nodes.

After completing the two CSV files containing nodes and edges of the pose graph, we started the optimisation process. For testing purposes, we prepared a MATLAB script to run the pose graph optimisation algorithm because MATLAB can display the result as charts and histograms for easy follow-up. The MATLAB script reads the extrinsic parameters data set in the two CSV files, with the support of the Optimisation Toolbox of MATLAB, calculating the most optimal relative pose from more than 400 transformations between each camera and board's poses. The optimisation results are demonstrated in the figures below.

Table 4.3: Pose Graph Optimisation result

Cam	t_x	t_y	t_z	q_x	q_y	q_z	q_w
7	0	0	0	0	0	0	1
8	-0.3963	-0.018545	0.0095207	-0.0070655	-0.0068763	0.0057706	0.99993
9	-0.79415	-0.025504	0.0021866	-0.0070058	-0.0088555	0.0090403	0.9999
10	-1.0534	-0.024433	0.17173	-0.023018	0.25748	0.0039374	0.966

Sum of residual error before optimising: 335.4759 (mm)
Sum of residual error after optimising: 16.90129 (mm)

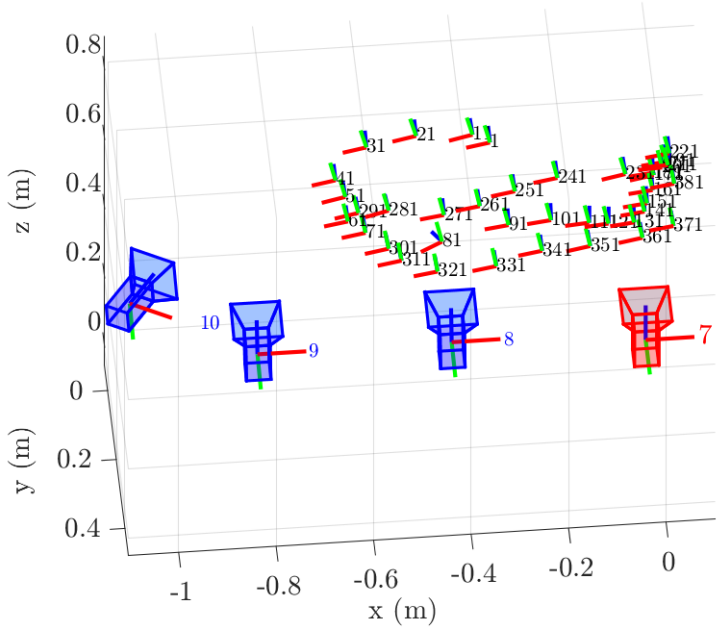

Figure 4.11: Poses of the optimised cameras and calibration board. Cam 7, which is coloured in red, is the origin. Every 10^{th} pose of the calibration board's origin represented by a 3D axis

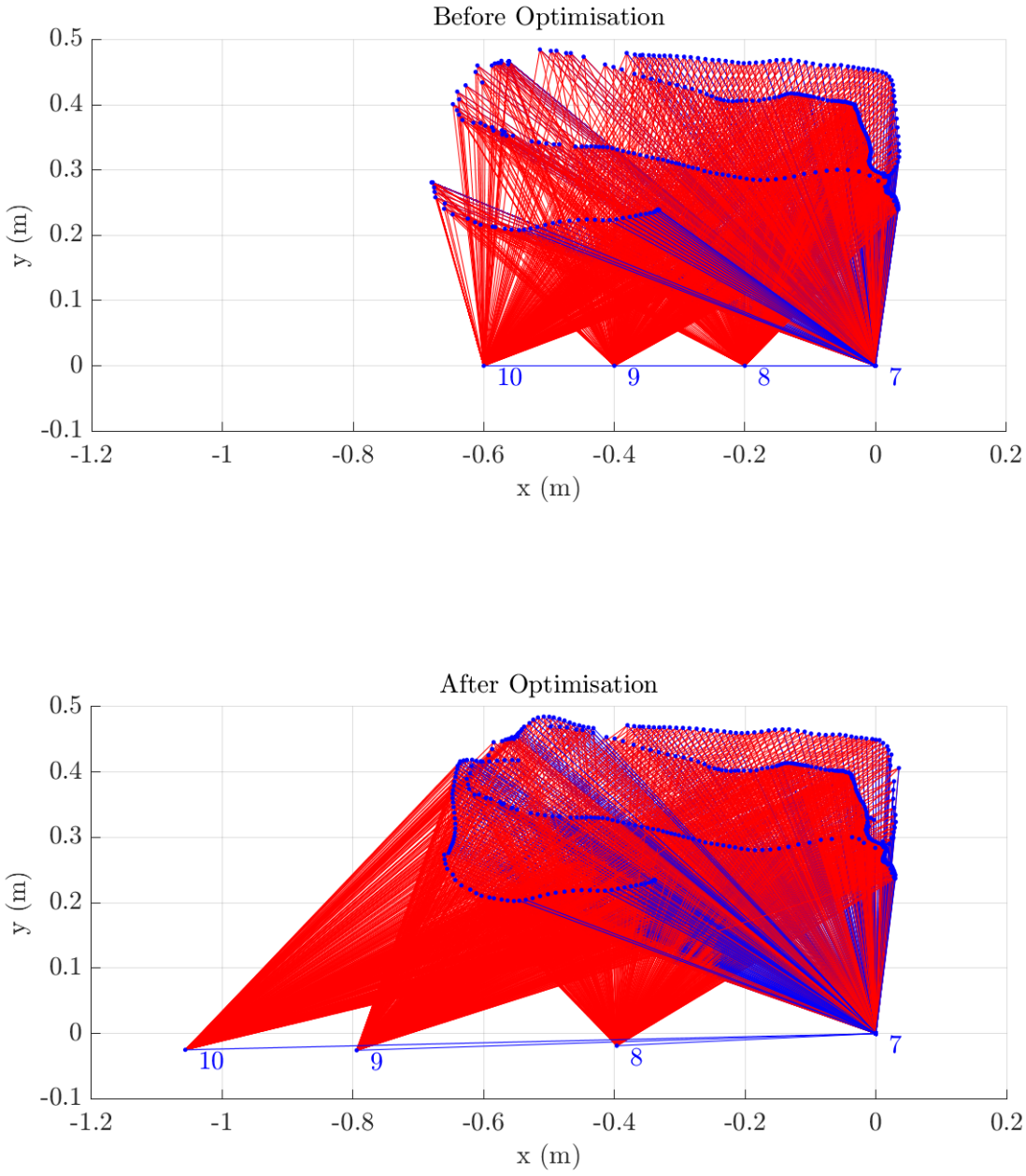

Figure 4.12: The pose graph before and after the optimisation is carried out. Blue circles are the nodes, red lines are the edges, and blue lines are loop-closure edges. The four camera nodes have been labelled in blue.

Chapter 5

Discussion

5.1 Accuracy

Our approach covered in this capstone report consists of two main parts: proposing a double-sided ArUco board as the calibration target and applying General Graph Optimisation (G2O) to enhance the accuracy of the calibration result.

In terms of the double-sided ArUco board, experiments have shown that this approach worked well in our project scope. All 16 cameras in our system could thoroughly detect the double-sided ArUco board in their field of view, and then our program framework could estimate the board's poses which are respected to each camera. Moreover, Open-CV library's solvePnP() function can calculate the transformation that minimises the re-projection errors and guarantees accuracy of the calibration results. To improve the calibration result with our approach, we suggest increasing the number of ArUco markers on both sides as well as their dimension; the more ArUco tags captured and detected by the camera, the higher the accuracy of the relative pose estimated.

On the other hand, the final extrinsic parameters are also calculated very accurately after finishing the pose graph optimisation process. The residual errors for total edges of the pose graph before optimising is 335.4759 millimetres, being significantly reduced to 16.90129 millimetres after optimising. Considering our pose graph has a total number of 1849 edges, the average residual error is just 0.009 millimetres per edge. This value can be regarded as an acceptable error. If possible,

we recommend increasing the number of nodes in the pose graph by capturing more images, and it will help enhance the result of the optimisation process.

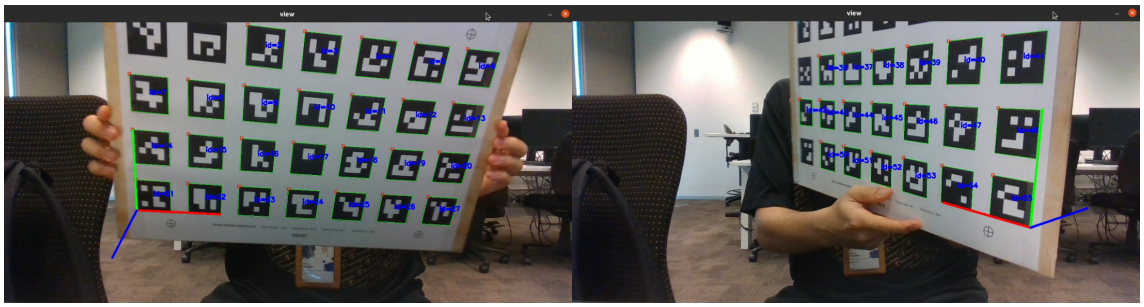

Figure 5.1: ArUco board estimation (both sides)

Figure 5.2: Edge Residual Errors by Edge ID

5.2 Further works

We have only demonstrated the calibration of four cameras, but the entire system contains sixteen. Complete calibration of all cameras requires synchronisation of images with the reference camera, which needs to be tested. Furthermore, increasing the number of cameras may affect the convergence of the optimisation and final accuracy of the calibration results. Testing will need to be carried out to exclude board poses with fewer edges, i.e. fewer cameras see the board in that pose.

Currently, there is no way of getting ground truth poses between the cameras with the physical camera system. Therefore, we can not quantitatively test the accuracy of the estimated poses from our method. It may be possible to set up a simulator with multiple cameras, where the pose between the cameras is known, to test the calibration procedure. Noise can be added to images to replicate the functionality of physical cameras. Our calibration approach could also be tested with existing methods to compare results.

An important quantity that has not been considered is the uncertainty of our calibration parameters arising from the propagation of sensor noise. The calibration method relies on detecting fiducial markers in noisy images of the calibration board. The corners, in pixel coordinates, that are used in the optimisation will have some associated uncertainty. This uncertainty will be carried through the estimation of the board poses, and through the graph-based optimisation. Estimating the uncertainty can help us to understand the reliability of the final calibrated parameters, and potentially guide the collection of new images from cameras with high uncertainty. In completing this task, there is a possibility to publish the findings to a research conference.

Additionally, we would like to integrate the Pose Graph Optimisation into our program framework, so we will not need to save the relative pose between the ArUco board and each camera to CSV files, then run the MATLAB script for the optimisation. The General Graph Optimisation (G2O) algorithm will be coded in C++ and directly integrated into our framework, and then the optimisation process will be automatically run after finishing the board detection process for each camera.

Besides, we would like to modify our program framework to be more modular to improve the processing speed and efficiency of the program. We will publish our framework in a public repository on GitHub and as an open-source ROS package for users.

Chapter 6

Conclusion

In conclusion, this capstone report proposes an approach for camera calibration of multiple partially overlapping cameras to determine the extrinsic parameters of the camera system. This project is a part of the "Objective real-time live cattle assessments to improve profit" funded project by Meat and Livestock Australia.

As a part of camera calibration, this report proposes to use an augmented planar board as the calibration target. A double-sided planar board with ArUco markers is employed to solve the problem with cameras facing each other and having overlapping fields of view. In addition, the General Graph Optimisation (G2O) is applied to optimise the calibration results. Experiments were also implemented to test the approach's feasibility, and the results obtained were also very encouraging. However, this method still needs improvement to increase efficiency and accuracy.

In the future, the program framework, as well as the algorithm, would be improved to test with the whole 16 cameras of this system. We would intend to publish our approach as a ROS package and store it on a public repository on GitHub for further development.

Bibliography

Baker, P. T. & Aloimonos, Y., 2003, 'Calibration of a Multicamera Network', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 7, p. 72. 4

Dong, S., Shao, X., Kang, X., Yang, F. & He, X., 2016, 'Extrinsic calibration of a non-overlapping camera network based on close-range photogrammetry', *Applied Optics*, vol. 55, no. 23, p. 6363. 5

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J. & Marín-Jiménez, M. J., 2014, 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292. 10

Ikeuchi, K., 2014, 'Computer vision: A reference guide', . 2

Korthals, T., Wolf, D., Rudolph, D., Hesse, M. & Ruckert, U., 2019, 'Fiducial marker based extrinsic camera calibration for a robot benchmarking platform', *2019 European Conference on Mobile Robots, ECMR 2019 - Proceedings*. 10

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. & Burgard, W., 2011, 'G2o: A general framework for graph optimization', *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3607–3613. 2

Marcon, M., Sarti, A. & Tubaro, S., 2017, 'Multicamera rig calibration by double-sided thick checkerboard', *IET Computer Vision*, vol. 11, no. 6, pp. 448–454. 7

Olson, E., 2011, 'AprilTag: A robust and flexible visual fiducial system', *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3400–3407. 10

Reinke, A., Camurri, M. & Semini, C., 2019, 'A Factor Graph Approach to Multi-camera Extrinsic Calibration on Legged Robots', *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pp. 391–394. 11

Shen, E. & Hornsey, R., 2011, 'Multi-camera network calibration with a non-planar target', *IEEE Sensors Journal*, vol. 11, no. 10, pp. 2356–2364. 9

Triggs, B., 1999, 'Camera pose and calibration from 4 or 5 known 3D points', *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, pp. 278–284. 5

Tsai, R. Y., 1987, 'A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses', *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344. 4

Ueshiba, T. & Tomita, F., 2003, 'Plane-based calibration algorithm for multi-camera systems via factorization of homography matrices', *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 966–973. 5

Yin, L., Wang, X., Ni, Y., Zhou, K. & Zhang, J., 2018, 'Extrinsic parameters calibration method of cameras with non-overlapping fields of view in airborne remote sensing', *Remote Sensing*, vol. 10, no. 8. 7

Zhang, Z., 2000, 'A flexible new technique for camera calibration', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334. 4