RGB-D Camera Network Calibration for 3D Model Reconstruction

João Gabriel Albuquerque Simões Furtado Ramiro

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Manuel Ricardo de Almeida Rodrigues Marques
Prof. João Paulo Salgado Arriscado Costeira

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Manuel Ricardo de Almeida Rodrigues Marques
Member of the Committee: Prof. Jacinto Carlos Marques Peixoto do Nascimento

November 2019

Declaration

Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração:

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Abstract

Accurate and real-time 3D reconstruction of objects is an aspiration that has seen many developments in the past years. Many approaches use a multi RGB-D camera setup and thus require a precise estimation of the relative poses between all cameras. This thesis presents an approach to calibrate a RGB-D camera network system using a known calibration object based on fiducial markers. Results show that the method yields an initialization that is close to the real poses. Afterwards, a joint multi-cloud registration method performs fine adjustments on the obtained poses to attain a more accurate outcome. We use the system to create a 3D face model that is then used to track the mouth of a person in an automatic human feeding application. The designed calibration framework is lightweight, flexible and scalable since it can be used in networks with several cameras, achieving full system calibration in a few minutes.

Keywords

Camera networks, 3D reconstruction, RGB-D data, Multi Cloud Registration

Resumo

A reconstrução de objetos 3D em tempo real é um objetivo para o qual se tem realizado investigação extensiva. Diversas abordagens recorrem a um sistema com múltiplas câmaras RGB-D, e que por isso necessita do conhecimento prévio das poses relativas entre câmaras. Nesta tese, apresenta-se um método de calibração de uma rede de câmaras RGB-D, utilizando um objeto de calibração conhecido baseado em marcadores fiduciais. Os resultados mostram que a abordagem implementado gera poses próximas da reais. Ainda assim, é utilizado um algoritmo para ajustes finos, fundamentado no registo simultâneo de nuvens de pontos, aproximando os resultados ainda mais à realidade. Este sistema é posteriormente utilizado para gerar modelos 3D, nomeadamente o de um rosto humano, com fim à integração numa aplicação automática para alimentar pessoas incapacitadas. O método em estudo mostra-se computacionalmente leve, flexível e escalável, uma vez que funciona com diferentes objetos de calibração e diferentes quantidades de câmaras, alcançando a calibração do sistema em poucos minutos.

Palavras Chave

Redes de Câmaras, Reconstrução 3D, Informação RGB-D, Registo de Múltiplas Nuvens de Pontos

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.2 Contributions . . . 3

1.3 Outline . . . 3

2 State of the Art **5**

2.1 Calibration of camera networks . . . 6

2.2 Point Cloud Registration Approaches . . . 8

2.3 Summary . . . 9

3 Object Transformations from Relative Poses **13**

3.1 Transformations from Relative Poses Problem Description . . . 14

3.2 Solution for the Rotations . . . 17

3.3 Solution for the Translations . . . 19

3.4 Calibration Method . . . 20

4 Camera Network Calibration **24**

4.1 Camera Pose Initialization . . . 25

4.1.1 Calibration Method . . . 26

4.2 Camera Pose Fine Adjustments . . . 26

4.2.1 General Procrustes Analysis . . . 27

4.2.2 Algorithm for multiple point cloud registration . . . 29

5 Experimental Evaluation **31**

5.1 Calibration Object Experiments . . . 32

5.1.1 Description of the Calibration Object . . . 32

5.1.2 Calibration Object Evaluation . . . 33

5.1.3 Alternative Calibration Objects . . . 35

5.2 Camera Poses Experiments . . . 37

5.2.1 Qualitative Experiments on Camera Poses . . . 37

5.2.2 Fine Tuning The results . . . 38

5.2.3 Error Analysis of the Camera Poses . . . 39
5.3 3D Data Fusion . . . 42
5.4 Limitations . . . 44
6 Application **45**
6.1 Description . . . 46
6.2 Results . . . 47
7 Conclusions & Future Work **50**
7.1 Conclusions . . . 51
7.2 Future Work . . . 51
A Appendix A: Depth Cameras **58**
B System Description and Functionalities **61**
B.1 Hardware . . . 62
B.2 Software . . . 62
B.2.1 Configurations . . . 63
B.2.2 System Start and Stop . . . 63
B.3 Functionalities . . . 63
B.3.1 System Calibration . . . 63
B.3.1.A Obtaining Calibration Object . . . 64
B.3.1.B Obtaining Camera Poses . . . 64
B.3.2 Real-Time 3D Reconstruction . . . 65
B.3.3 Point Cloud Cropping . . . 65
B.4 Limitations . . . 65
B.5 Scalability and Flexibility . . . 66

List of Figures

1.1 Acquisition of the 3D model of a robotic arm using the system developed. 3

3.1 Object Transformations. 14
3.2 Known Object. 16
3.3 Pairwise transformations within calibration object. 17
3.4 Marker Pose Estimation Flowchart. 21
3.5 Obtained Marker Poses. 22
3.6 Marker axis orientation. 23
3.7 Scenes. 23

4.1 Object Transformations. 25
4.2 Camera Pose Estimation Flowchart. 26
4.3 Point matching. Images taken from [7]. 29
4.4 Global Procrustes Analysis ICP (GPA-ICP) Before vs After 30

5.1 Calibration Object 32
5.2 Calibration Object vs Model comparison 33
5.3 Average Reprojection Error in one of the experiments. 34
5.4 Average Reprojection Error in one of the experiments. 35
5.5 Detections and reprojections comparison. 36
5.6 Cube Calibration Object. 36
5.7 Scene Calibration Object. 37
5.8 Camera Poses in Test 1. 37
5.9 Camera Poses in Test 2. 38
5.10 Fine tuning when without initialization. 38
5.11 Fine tuning using initialization from test 1. 39
5.12 Test 3: Camera Poses Pointing at an robot arm. 39
5.13 Scene 3 Before vs After GPA-ICP. 41

5.14 Before vs After GPA-ICP Part 1. 42
5.15 Before vs After GPA-ICP Part 2. 43

6.1 Feedbot Feeding System. 46
6.2 3D Face Model Obtained. 48
6.3 Discriminative Optimization Results. 48
6.4 Partial face detection with DO. 48
6.5 Computer vizualisation of the Feedbot system. 49

A.1 Pinhole Camera Model 59
A.2 Types of Depth Cameras. Taken from[1]. 60

B.1 Utilized 3D Camera. 62
B.2 System Diagram. 63
B.3 Calibration Object Calibration Setup. 64

List of Tables

2.1 Camera Networks Implementation Summary. 11

5.1 Diagonal Measurements. 34
5.2 Reprojection Error Statistics [px]. 35
5.3 Camera Pose Errors for 3 initializations before and after GPA-ICP. 40
5.4 Effect of different scenes in GPA-ICP performance. 41

Acronyms

LS Least Squares

TLS Total Least Squares

PnP Perspective-n-Point

ADL Activities of Daily Living

ICP Iterative Closest Point

GPA Generalized Procrustes Analysis

JR-MPC Joint Registration of Multiple Point Clouds

GMM Gaussian Mixture Model

ECM Expectation Conditional Maximization

BA Bundle Adjustment

SVD Singular value decomposition

GPA-ICP Global Procrustes Analysis ICP

DO Discriminative Optimization

SfM Structure from Motion

MLE Maximum Likelihood Estimator

ROS Robot Operating System

1

Introduction

Contents

1.1 Motivation 2

1.2 Contributions 3

1.3 Outline 3

1.1 Motivation

Since the beginning of civilization having a good point of view of our world has benefited humanity as a whole. For instance, in ancient Greece, Plato wrote the Allegory of the Cave [1], where initially prisoners were shackled inside a cave where all they could see was projections of the real world. When released they got to see the actual reality from another, clearer, perspective, thus gaining a better understanding of it.

We as a society have come a long way since then, creating instruments to measure and analyse everything around us. One of the most common ways of capturing what surrounds us is through images, initially gray scale, now in color. With times this technology has evolved reaching the point where we are today, depth images, from which we are able to obtain the 3D depiction of the world.

Since the launch Microsoft Kinect [2], RGB-D cameras, which retrieve both color and depth images have gained popularity leading to many consumer grade 3D cameras, being released, leading to a more affordable sensor. Such tool has then seen many uses in the most varied areas, due to its particular ability to identify, locate and track objects in its view [3].

To obtain 3D data of large areas and get a more complete representation of objects, it is possible to combine multiple RGB-D sensors to record the 3D reconstructions of a scene. Consequently, it is possible, for instance, to track objects in space over time which is important in areas such as in manufacturing, teleimmersion, surveillance and even healthcare [4].

Specifically, in this work we propose a method to calibrate a network of RGB-D cameras. To calibrate means to acquire the camera positions and orientations relative to all cameras. The calibration enables us to 3D reconstruct the surrounding space such as in Fig. 1.1 , that can then be used to perform tasks in a more controlled and precise manner. One of the primary objectives of this thesis is to use the 3D reconstructions to track objects in a multiple camera system. Particularly we apply our 3D reconstructions to Feedbot [5, 6], a robot arm application that feeds meals to people with lack of upper-body mobility.

The proposed goal of calibrating a camera network contains two steps. In the first one, using a known object, we obtain an initialization of the camera poses by detecting the known object on the RGB images, and solving a Least Squares Problem. The second step consists in using the 3D reconstructions originated from each camera to further improve the estimated camera poses, using a global optimization methodology named Global Procrustes Analysis ICP (GPA-ICP) which is described in [7], to register point clouds simultaneously onto the same referential. We also developed a physical system in order to test our approach. This system is based on Intel ZR300 depth cameras that use Robot Operating System (ROS) to transmit data into a main server where most of the computations are done.

The whole process yields accurate results, is flexible and can be applied to both large and small 3D camera networks to produce 3D reconstructions of scenes and objects, that can then be used to

Figure 1.1: Acquisition of the 3D model of a robotic arm using the system developed.

generate 3D models and to track objects across time.

1.2 Contributions

This thesis has the following main contributions:

- **Multi RGB-D Camera Network Calibration Method:**
  We supply a method that uses marker based RGB feature matching in multiple cameras, followed by a joint multiple point cloud registration method, to obtain the global poses of the full camera system. Its ROS and python packages will be available in this repository[1].

- **Acquisition of More Complete Models:**
  We provide a method to stream in real-time a 3D reconstruction of a scenario, using RGB-D cameras spread around it. Alternatively, the cameras can also be placed in such a way that full 3D models of faces and objects can be obtained.

1.3 Outline

This thesis is partitioned into 7 chapters. Chapter 1 contains the introduction for this thesis, which includes the motivation behind it, the objectives for it, and its outline. Chapter 2 contains a review of the recent progress in the area, which covers various approaches for the camera network calibration problem, and also on the registration of multiple point clouds. In chapter 3 we describe how to tackle the main problem and formulate it mathematically, reaching an optimization problem that is then decoupled into two independently solvable problems. In chapter 4 we reveal how the developed method can

[1] https://github.com/DonHaul/MscThesis

calibrate a camera network using the aforementioned formulation and a joint point cloud registration technique. In chapter 5, we show and interpret the calibration results and validate the designed approach. Chapter 6 shows how the 3D reconstructions obtained can be used in a real world face tracking application. In chapter 7, conclusions are drawn, and future work is proposed to further improve and extend this solution.

In the Appendix A a description of the pinhole camera model is done as well as overview on the technologies behind 3D cameras. Appendix B describes the camera network system to its full extent, both hardware and software wise, presenting its limitations and functionalities.

2

State of the Art

Contents

2.1 Calibration of camera networks 6

2.2 Point Cloud Registration Approaches 8

2.3 Summary 9

2.1 Calibration of camera networks

Many camera sensors networks with different architectures have been developed over the years and, consequently, multiple calibration approaches have come with them.

Some systems such as [8–10] offer very precise integrated solutions but at great cost. This is not the objective of this study, for which this chapter comprises a review of related work that present fewer, simpler and less expensive cameras networks.

To obtain relations between cameras many works use a calibration object. The authors in [11] use a lighted sphere which is placed within the view of multiple cameras. Images are taken with the ball position in at least 3 noncollinear locations. Since they use a lighted sphere, it is easy to detect it in each image just by thresholding, from which the sphere's center can be determined. They then use Orthogonal Procrustes Analysis to register the cameras in a pairwise manner by using the estimated center spheres. After an initial pose estimation is obtained, an optimization problem that jointly registers and refines the whole cameras poses is used. The work in [12] also uses spherical objects to perform calibration. Each sensor detects known spheres, segments them and estimates their center points. The extrinsics are then calculated using Singular value decomposition (SVD) followed by a final optimization using Maximum Likelihood Estimator (MLE) that refines the detected sphere centers and consequently the extrinsic parameters of the network.

The authors from [13, 14] also use a sphere to calibrate the extrinsics of an RGB-D camera network. Their approach is interesting since instead of modeling the camera extrinsic calibration using a rigid transformation, optimal only for pinhole cameras, i.e. ideal cameras, they test different view transform functions. Finally they utilize Bundle Adjustment (BA) to fine-tune the initial calibration estimates.

An alternative to spheres are virtual calibration objects which are used in [15–17]. The authors from [15] use two moving LED markers to generate a virtual calibration object in 3D space. After several images are acquired, the cameras are pairwise registered with the use of epipolar geometry. A graph is produced describing how cameras are related to each other by measuring the parital overlap in their views. BA is used to refine the final poses. In [16] a laser pointer is used to make a bright spot appear in the view of each camera. They epipolar geometry to estimate the projective depth followed by rank-4 factorization to further compute the projective structures. In the end, they compute the extrinsic and intrinsic parameters of the camera via Euclidean stratification and refine them using BA.

The study in [18] proposes a method to calibrate multiple depth cameras that uses a large plane to generate virtual points is proposed. The equation of this plane is calculated in each image at every time instant, by manually selecting the four plane corners. Finally, by intersecting three planes from different time instants a virtual point is generated. By repeating this process a virtual point cloud is generated with matches across all cameras, which allows them to obtain the pose between pairs of cameras using Procrustes analysis. BA is used in the end to enhance the results.

In [17] a structure for motion approach was taken to get poses between camera pairs. This poses are then incrementally added onto a single reference frame, where a transformation from one camera to any other is known. An iterative refinement is done after that.

The works in [19–22] use people as means to calibrate the network, some make usage of their movement in the room and others of their still poses. In OpenPTrack [19] they calibrate an RGB-D camera network by initially moving a checkerboard in front of the cameras. At each time instant that the checkerboard is detected by two or more cameras, their relative pose is added to a global pose graph. This process is repeated until a connected pose graph is obtained. After that, a special calibration toolkit described in [23] is used to solve the constrained optimization problem at hand. As a final step, a person moves around the scene, and each camera generates tracklets of his/her positions, that are then merged througth an iterative process to refine the obtained camera poses. The first part of these works is very similar to ours, as it uses a known calibration object to obtain relations between pairs of cameras. However, it can only be used in setups where multiple cameras are observing the scene from similar points of view, as the checkerboard that was used only has identifiable features in one of the sides, which does not occur in our method, since the used calibration object has features all around it, enabling for it to be identified no matter the points of view.

In [20] the authors also calibrate the system using the paths that pedestrians go through. Firstly a method is proposed to accurately predict the 3D positions of the person's head, using RGB images only. Then a RANSAC-based orthogonal Procrustes is used to get pairwise extrinsic parameters between cameras. The final step refines this extrinsic calibration by minimizing a objective function that regards the reprojection error.

In [21], no calibrations objects are used, instead, motion features are extracted by having people walking around the scene. Afterwards a Hough Transform is applied to define the geometry between camera pairs without the need to find matches. To globally register the camera network, cameras are successively added into a pose graph. Because the final calibration result depends on the order the cameras are added to the graph, multiple combinations are experimented and evaluated using a global optimization criterion.

The authors from [24] take a slightly different approach by using a pedestrian with known height. The height of the camera is estimated and it is assumed that the camera has no roll, thus only the pitch and the yaw are estimated. A global optimization is then performed, by using a planar trajectory alignment algorithm to align the person's trajectory. As a final refinement an adaptive chaos particle swarm optimization is executed.

Another alternative is the use of fiducial markers, which are easily identifiable objects, usually planar, that serve as a point of reference. This markers are used in [25] to obtain an initial rough estimate of the cameras pose in a world reference frame. They assume that the fiducial marker pose is known so as

to greatly facilitate the process. Because they use 3D cameras, they have access to the point clouds of each camera, which are used to perfect the results using Iterative Closest Point (ICP). A type of fiducial markers are also utilized in [26] and [27] that both use ArUco markers to obtain an initial rough estimate of the poses. Using the 3D data from the cameras, the calibration is refined using cropped point clouds corresponding to volumes where ArUco markers are detected by incrementally joining them through ICP. The method we developed has some similarities with [25–27], as it first obtains a rough estimate of the poses using markers, and then uses the 3D data to improve the resuts. The difference is that, while in the referred works the marker poses are static and are already manually registered in a world reference frame, in this thesis the markers make part of a calibration object which moves and that is not registered manually.

In [28] the authors create their own algorithm, BAICP+, which combines BA the ICP algorithm taking into account 2D and 3D information from the scene. Their technique requires a considerable amount of overlap between cameras, as well as a double-sided chessboard during the calibration process.

In [29], a network with RGB, RGB-D and Lidars is used. The authors perform a joint calibration by first grouping some of the sensors and etching poses within each group using an optmization problem with geometric constraints such as distance preservation, collinearity and coplanarity. To estimate poses between different groups they use Horn's algorthm or Levenberg-Marquardt optimization. A relevant detail is that one sensor may be included in multiple groups in order to add robustness to the single objective function that is used to calibrate whole system.

The authors of [30] propose a coarse-to-fine registration method. The initial rough calibration is conducted by making use of the virtual object calibration methods described in [15, 16], after that they take advantage of the algorithm in [7] to better align the point clouds. In the end they perform BA to further refine the obtained poses.

In [31], a Massive Multiview System was built using 480 RGB cameras, and 10 RGB-D sensors. Structure from Motion (SfM) is first used to calibrate both the extrinsic and intrinsic parameters of the RGB cameras. The process is followed by BA to refine the results. The RGB-D cameras are calibrated using a chessboard to get the homographies.

2.2 Point Cloud Registration Approaches

As described by some of the methods above it is possible to use the 3D data, i.e. point clouds, from different cameras/points of view to estimate poses between those points of view. One of the most known point set registration methods is ICP [32], an iterative method, that at each iteration finds correspondences between points in different clouds to approximate one point cloud to the other. Since ICP was first developed, a lot of progress has been made towards capturing tridimensional, and with that

a higher demand for accurate and flexible method capable of multiple point cloud registration method.

The system developed in [33] uses a stationary Lidar that over a 3 minute duration scans the surrounding area, the process is then repeated multiple times by positioning the sensor in different locations, the obtained point clouds are then registered using both manually, and automatically using feature matching to provide an initialization for ICP, to align pairs of point clouds. This system creates an accurate 3D reconstruction of the area, but without recording capabilities.

As mentioned above, ICP can only be used to register pairs of point clouds, so if we wanted to align multiple point clouds, we would have to use it incrementally which would generate error with each point cloud that is registered. An alternative to it are global registration methods that align multiple point clouds at once. One of this methods is explained in [34], where a global registration is performed to merge several underwater range images. Due to noisy and low resolution point clouds, the authors use a global registration method approach so that the error is evenly distributed across all views. First, an initial pairwise registration is performed using ICP. For this step, they need to know which views overlap, i.e. have similar point cloud structure. This is a limitation because in systems such as ours, there is no sequence information, since what we have is not one moving camera but multiple static ones. The global registration is performed by optimizating a non-linear Least Squares (LS) function, that is minimized using the Gauss-Newton algorithm.

The authors of [7] take advantage of Generalized Procrustes Analysis (GPA), which is a well-known technique that provides a least-squares solution to the multiple point set registration problem, and embed it in a ICP Framework, thus creating the Global Procrustes Analysis ICP (GPA-ICP) algorithm. This algorithm registers, i.e. transforms, point clouds that have similarities between them, so that in the end they better match each other. This will be further described in chapter 4, since this algorithm is used in the method developed in this thesis.

The Joint Registration of Multiple Point Clouds (JR-MPC) algorithm is proposed by the authors in [35]. In this algorithm each point cloud is the realization of a Gaussian Mixture Model (GMM), thus casting the problem into a clustering problem, where by finding the parameters of the GMM the poses of the point clouds are also found. The formulated optimization problem is solved using Expectation Conditional Maximization (ECM). A hierarchical GMM implementation is presented in [36], where the point matching of different point clouds, is done over multiple scales as a recursive tree-based search thus increasing efficiency on point matching. The result is a fast and accurate point cloud registration.

2.3 Summary

Most of the reviewed works utilize some sort of calibration object, which facilitates the camera network calibration process, because there is a known object that can be easily detected by every camera. The

developed work also uses a calibration object based on fiducial markers. A downside of not having a calibration object is that it may not be able to replicate the same calibration process in other scenarios. Some of the works that do not use a calibration object, have to use keypoint feature matching instead, which will only be useful in feature rich environments.

Furthermore, despite the majority of the works analyzed having some form of joint calibration of the multiple views, only some of them use this methodology since the beginning ([11, 12, 15, 17, 28, 30, 31]), the rest of them have an initialization based on pairwise registrations of cameras, and only after that the global registration is performed. Our approach also uses pairwise registrations to set an initialization for the camera poses but it combines them all into the same one optimization problem so that a global registration is done from the get-go. Moreover some RGB-D camera networks [19, 25–27] use ICP to merge the point clouds from each camera thus refining the results. This is not ideal, as it will accumulate error in each pairwise registration. By using GPA-ICP, [30] and our method aligns the point clouds globally without propagating the error.

Finally our method uses RGB-D cameras as most of the state of the art, which are cheap consumer-grade cameras. Although the built system may not be as precise as full-blown costly systems such as [8–10], we provide a simple and cheap way of obtaining a calibrated system capable of having a 3D representation of the scene.

The type of sensors, and calibration techniques used in the reviewed work and on the proposed method are summarized in Table 2.1.

Table 2.1: Camera Networks Implementation Summary.

Source	Type of Sensors	Calibration Tool	Joint Registration	Technique Used
[11]	RGB	Sphere	Yes	rank-4 factorization,Procrustes,Gradient Descent
[12]	RGB-D	Three Spheres	Yes	MLE , SVD
[13]	RGB-D	Sphere	Yes	SVD, BA
[15]	RGB	LED markers	Yes	RANSAC, Dijkstra BA
[17]	RGB	-	Yes	SfM, Extended Kalman FIlter
[16]	RGB	Bright Spot	Yes	RANSAC, rank-4 factorization, BA
[18]	RGB-D	Large Plane	Yes	Procrustes Analysis, BA
[19]	RGB-D	Checkerboard, Pedestrians	Yes	Constrained Optimization, ICP
[20]	RGB	Pedestrian Trajectories	Yes	Orthogonal Procrustes, RANSAC, Gradient Descent
[21]	RGB	Pedestrians' Motion	Yes	Hough Transform
[24]	RGB	Pedestrians' Head	Yes	Adaptive Chaos Particle Swarm Optimization
[25]	RGB-D	Fiducial Markers	No	Orthogonal Procrustes, ICP
[26]	RGB-D	Fiducial Markers	No	Perspective-n-Point (PnP), ICP
[27]	RGB-D	Fiducial Markers	No	Quaternion Averaging, ICP
[28]	RGB-D	Double-Sided Chessboards	Yes	BAICP+
[29]	RGB + RGB-D	Checkerboard, Plane	Yes	MLE, Horns Algorithm
[30]	RGB-D	Checkerboard	Yes	Zhang's Method, GPA-ICP , BA
[31]	RGB + RGB-D	Checkerboard, Textured Tent	Yes	sfm, bundle adjustment
Proposed	RGB-D	Fiducial Markers	Yes	Total Least Squares (TLS), LS, Procrustes Problem, GPA-ICP

3

Object Transformations from Relative Poses

Contents

3.1 Transformations from Relative Poses Problem Description 14

3.2 Solution for the Rotations 17

3.3 Solution for the Translations 19

3.4 Calibration Method 20

In this chapter we formulate the problem of obtaining the transformations of objects in a common reference when only pairwise transformations between objects are initially known. We then tackle the problem by decoupling it, into two separate simpler problems, which we show how to solve. In the end this process is applied to a real use case that is relevant to this thesis, which is to obtain a model of the calibration object.

3.1 Transformations from Relative Poses Problem Description

The main problem we want to address here is the calibration of a camera network, i.e. we want a way of knowing the transformations of a camera, in relation to every other camera in the system. The aforementioned problem can also be generalized into the following: Having a set of objects where we know the transformations between pairs of elements, how can we convert those object pairwise transformations (Fig. 3.1a) into transformations relative to a single reference frame (Fig. 3.1b)?

If we consider that every object is a node, then each edge would correspond to the known transformation between each pair of objects, thus forming a graph. For the system to be calibrated this graph must be a connected graph, i.e. a graph where every node has a path to every other node. We can see this in Fig. 3.1 where there are paths connecting all objects C_i together.

(a) Pairwise Transformations. **(b)** Global Transformations.

Figure 3.1: Object Transformations.

The transformations we refer to are 3D rigid transformations, which contain a translation and a rotation component. They can be represented by

$$\overbrace{\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}}^{x_1} = {}^1R_2 \overbrace{\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}}^{x_2} + {}^1t_2, \tag{3.1}$$

where ${}^1R_2$ is a 3×3 rotation matrix and ${}^1t_2$ a 3×1 translation vector that transform some point

$\boldsymbol{x_2}$ in reference frame 2 into $\boldsymbol{x_1}$ in reference frame 1. If there are two point sets X_1 and X_2 and the correspondences between points in each set are known, we are able to find the transformation in Eq.(3.1) by solving the Procrustes Problem [37] which boils down to solving

$$\begin{aligned} \hat{R}, \hat{t} = \quad & \underset{R,\, t}{\arg\min} \quad ||X_1 - \big(RX_2 + t\big)||_F^2 \\ & \text{subject to} \;\; R \in SO(3), \end{aligned} \tag{3.2}$$

where SO(3) represents the set of 3D rotation matrices characterized by $R^T R = 1$ and det(R)=1.

However, if those correspondences are not known, the problem is transformed into Eq.(3.3), where we wish to obtain the transformation $\hat{R}$ and $\hat{t}$ between point sets X_1 and X_2 with P points each, α_{kl} defines the matching between points k and l. X_{1k} and X_{2l} represent the kth from point set X_1 and the lth of set X_1. Since there are many techniques and variants to match points, the generic constraint is added $\mathcal{F}(\alpha_{kl}) = 0$, which represents some of those variants, e.g. if a point has multiples matches, if the matches are weighted, or how a match is even defined.

$$\begin{aligned} \hat{\alpha}_{kl}, \hat{R}, \hat{t} = \quad & \underset{\alpha_{kl},\, R,\, t}{\arg\min} \quad \sum_{k,l}^{P} \alpha_{kl} ||X_{1k} - \big(RX_{2l} + t\big)||_F^2 \\ & \text{subject to} \;\; R \in SO(3), \\ & \qquad\qquad\quad k = 1, ..., P \quad l = 1, ..., P, \\ & \qquad\qquad\quad \mathcal{F}(\alpha_{kl}) = 0 \end{aligned} \tag{3.3}$$

This problem is inherently hard since it has constraints, but it can be approximated through the Iterative Closest Point (ICP) algorithm, which defines as match points that are the nearest neighbor to each other. This approach may or may not converge to the correct minima, depending on the point sets.

It is however more complicated when there are multiple point sets, from which we want to know the transformation in one global reference frame, the world. This is described by the optimization problem in Eq.(3.4),

$$\begin{aligned} \hat{\alpha}_{klij}, {}^{w}\hat{R}_i, {}^{w}\hat{t}_i = \quad & \underset{\alpha_{klij},\, {}^{w}R_i,\, {}^{w}t_i}{\arg\min} \quad \sum_{i,j}^{N} \sum_{k,l}^{P} \alpha_{klij} ||\, {}^{w}R_i X_{ik} + {}^{w}t_i - \big({}^{w}R_j X_{jl} + {}^{w}t_j\big)||_F^2 \\ & \text{subject to} \quad {}^{w}R_i \in SO(3), \\ & \qquad\qquad\quad i = 1, ..., N \quad j = 1, ..., N \quad k = 1, ..., P \quad l = 1, ..., P, \\ & \qquad\qquad\quad \mathcal{F}(\alpha_{klij}) = 0, \end{aligned} \tag{3.4}$$

where N is the number of sets, P the number of points, ${}^{w}R_i$ and ${}^{w}t_i$ are the rotation and translation from each point set i into the world frame. X_{ik} is ith point of set k. α_{klij} defines whether or not there is a correspondence between two points. More specifically, it defines the correspondence between point k from set i and point i from set j. This correspondence variable may have some constraints, portrayed

by $\mathcal{F}(\alpha_{klij}) = 0$, which as in Eq.(3.3), can define for example. e.g. if a point has multiples matches vs just one or if points are only a match when they are closer than a certain distance.

As we can see, Eq.(3.4) is a non-convex problem due to its many constraints, making it a hard problem as stated by some of the works described in Chapter 2.

In this work we have also developed a method to tackle this hard problem. We use a known object exposed in Fig. 3.2, that contains a set of identifiable markers, where we know their corner point positions, that can be easily found across all cameras. This means that variable α_{klij}, no longer needs to be found as it is previously determined by matching known points. By moving this object, of which we call of calibration object, in front of the cameras, it is possible through point correspondence between cameras to generate pairwise transformations, i.e. transformations between two cameras, which means that over time, rotations ${}^{j}R_i$ and translations ${}^{j}t_i$ between cameras i and j are acquired. This brings up the question: How can we use these transformations, that due to noise might contradict each other, to register all cameras on the same reference frame?

As we know, in many rigid transformations problems such as this one and the Procrustes Problem, the rotations can be found independently of the translations [37]. Because of this property this problem can be decoupled into a rotation and a translation problem described in Eq.(3.5) and Eq.(3.6).

$$\begin{aligned} {}^{1}\hat{R}_w, {}^{2}\hat{R}_w, ..., {}^{N}\hat{R}_w = \underset{{}^{i}R_w}{\arg\min} \quad & \sum_{i,j}^{N} || {}^{i}\hat{R}_w - {}^{i}\hat{R}_j \, {}^{j}\hat{R}_w ||_F^2 \\ \text{subject to} \quad & {}^{i}R_w \in SO(3), \\ & i = 1, ..., N \quad j = 1, ..., N \end{aligned} \tag{3.5}$$

$$\begin{aligned} {}^{1}\hat{t}_w, {}^{2}\hat{t}_w, ..., {}^{N}\hat{t}_w = \underset{{}^{i}t_w}{\arg\min} \quad & \sum_{i,j}^{N} || {}^{i}t_w - \left({}^{i}R_j \, {}^{j}t_w + {}^{i}t_j \right) ||_F^2 \\ \text{subject to} \quad & i = 1, ..., N \quad j = 1, ..., N \end{aligned} \tag{3.6}$$

Figure 3.2: Known Object.

The inputs in Eq.(3.5) and Eq.(3.6) are ${}^{i}R_j$ and ${}^{i}t_j$ which correspond to the pairwise rotation and translation between the objects with reference frames i and j. The minimization problem formulated for the rotations has a constraint and for that it is a non-convex problem, while the translations problem has a known format, the sum of the Euclidean norms, which can be solved with the Least Squares (LS)

method.

Before further describing the solution and obtaining the camera transformations relative to a unique reference frame, we first need a well-known calibration object so that we know correspondeces between cameras.

This object needs to have easily identifiable features all around so that multiple cameras from can identify it no matter what direction it is facing. The object in Fig. 3.2 is used in our method as it contains virtual reality markers, ArUco markers, that are easily detectable. To get its model, the transformations between markers must be known relative to each other. This will result in the problem stated before: Having a set of pairwise transformations between marker, how can we have their transformation relative to a single reference frame? Being the same problem we can solve it with the same method, in order to obtain every marker in the same world reference frame. Pairwise transformations are obtained as visualized in Fig. 3.3.

Figure 3.3: Pairwise transformations within calibration object.

Having this pairwise transformations it is now imperative to solve the optimization problems in Eq.(3.5) and Eq.(3.6). The former is solved in Section 3.2 while the latter is in Section 3.3.

3.2 Solution for the Rotations

In order to estimate the poses which consist of rotations R and translations t, we first estimate the rotations by manipulatiog the problem stated in Eq.(3.5). This is a non-convex problem due to its constraint. This constraint states that the obtained rotations ${}^{i}R_{w}$ must be contained in the $SO(3)$, which is the special orthogonal group in dimension 3, also known as the rotation group which states that $det(R) = 1$ and $R^T R = 1$. If we lift the constraint and are left with Eq.(3.7). This turns a rather complicated problem into a simple one.

$$ {}^{1}\hat{R}_w, {}^{2}\hat{R}_w, ..., {}^{N}\hat{R}_w = \underset{{}^{i}R_w}{\arg\min} \quad \sum_{i,j}^{N} || {}^{i}R_w - {}^{i}\hat{R}_j \, {}^{j}R_w ||_F^2 \tag{3.7} $$

The main term that we want to minimize is ${}^{i}R_w - {}^{i}\hat{R}_j \, {}^{j}R_w$, this means we want to it to be equal to 0, which creates the following equation:

$$ {}^{i}R_w = {}^{i}R_j \, {}^{j}R_w. \tag{3.8} $$

This is the equation we want to solve to obtain ${}^{i}R_w$, $i = 1, 2, ..., N$. That means we have a system of linearly independent equations with N unknowns where the number of the equations in the system is defined by the number of known pairwise rotations K. The referred system can be formulated in a matricial form as in Eq.(3.9).

$$ \underbrace{\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ I & 0 & 0 & \dots & 0 \\ 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ & \vdots & & \vdots & \end{bmatrix}}_{I_0} \underbrace{\begin{bmatrix} {}^{1}R_w \\ {}^{2}R_w \\ {}^{3}R_w \\ \vdots \\ {}^{N}R_w \end{bmatrix}}_{Z} = \underbrace{\begin{bmatrix} 0 & 0 & {}^{2}R_3 & \dots & 0 \\ 0 & 0 & 0 & \dots & {}^{1}R_N \\ {}^{3}R_1 & 0 & 0 & \dots & 0 \\ 0 & {}^{3}R_2 & 0 & \dots & 0 \\ & \vdots & & \vdots & \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} {}^{1}R_w \\ {}^{2}R_w \\ {}^{3}R_w \\ \vdots \\ {}^{N}R_w \end{bmatrix}}_{Z} \tag{3.9} $$

This matrices encode each equation of the system in each row. I_0 and E are $3K \times 3N$ matrices. Moreover, for every rotation pair ${}^{j}R_i$, a 3×3 identity matrix is added on the $3 \times j^{th}$ of I_0 and rotation itself is added on the $3 \times i^{th}$ column of E , all the other elements are 0. The matrices can then be rearranged into Eq.(3.10),

$$ (I_0 - E)Z = 0. \tag{3.10} $$

Because the system is overdetermined, i.e. $N > K$ on most use cases, and because noise will make some rotations contradict each other e.g. ${}^{2}R_3 \neq {}^{3}R_4 \, {}^{4}R_2$, there is no a solution Z that satisfies Eq.(3.10). In cases like this, the best solution will be one that approximates the null space of $I_0 - E$ shown in

$$ \begin{aligned} \hat{Z} = \underset{Z}{\arg\min} \quad & \sum_{i,j}^{N} ||(I_0 - E)Z||_F^2 \\ \text{subject to} \quad & \text{rank}(Z) = 3. \end{aligned} \tag{3.11} $$

This minimization problem can be solved by performing Singular value decomposition (SVD) on $I_0 - E$ resulting in $I_0 - E = U\Sigma V^T$, where the columns of U contain the left singular vectors of $I_0 - E$, the columns of V contain the right singular values, and Σ is a diagonal matrix with the singular values.

Then we extract the columns of V that have the least influence on generating $I_0 - E$, which will be the last three, which give us $\hat{Z}$, the best approximation of the null space.

Because we relaxed the problem by lifting the constraint in Eq.(3.7), the individual 3×3 matrices that make up matrix Z in Eq.(3.12) will not correspond to 3D rotation matrices.

$$\hat{Z} = \begin{bmatrix} {}^{1}\tilde{R}_w \\ \hline {}^{2}\tilde{R}_w \\ \hline {}^{3}\tilde{R}_w \\ \vdots \\ {}^{N}\tilde{R}_w \end{bmatrix} \tag{3.12}$$

Solving the orthogonal Procrustes problem in Eq.(3.13), where A and B are rotation matrices it is possible to project each 3×3 matrix ${}^{i}\tilde{R}_w$ onto the rotation matrix space, as shown in Eq.(3.14), for $i = 1, 2, \ldots N$.

$$\begin{aligned} &\underset{R}{\arg\min} \quad ||AR - B||_F^2 \\ &\text{subject to} \quad R^T R = I \end{aligned} \tag{3.13}$$

$$\begin{aligned} {}^{i}R_w = \quad &\underset{{}^{w}R_i}{\arg\min} \quad ||\,{}^{w}R_i\,{}^{i}\tilde{R}_w - I||_F^2 \\ &\text{subject to} \quad {}^{i}R_w{}^{T}\,{}^{i}R_w = I, \end{aligned} \tag{3.14}$$

This ${}^{i}R_w$ can then be used to calculate the translations relative to a world frame in Section 3.3.

3.3 Solution for the Translations

As previously stated, Eq.(3.6) can be solved using the LS method in order to obtain the global translations between objects. This is done using the pairwise translations between them as input in conjunction with the rotations from Section 3.2. The following formula that defines pairwise translation can be extracted from Eq.(3.6):

$${}^{i}t_w = {}^{i}R_j\,{}^{j}t_w + {}^{i}t_j. \tag{3.15}$$

Through mathematic manipulation and some of the 3D transformation properties it is possible to reach Eq.(3.16c).

$${}^{i}R_w\,{}^{w}t_i = {}^{i}R_j\,{}^{j}R_w\,{}^{w}t_j + {}^{i}R_j\,{}^{j}t_i \tag{3.16a}$$

$${}^{w}t_i = {}^{w}R_i\,{}^{i}R_j\,{}^{j}R_w\,{}^{w}t_j + {}^{w}R_i\,{}^{i}R_j\,{}^{j}t_i \tag{3.16b}$$

$${}^{w}t_i = {}^{w}t_j + {}^{w}R_j\,{}^{j}t_i \tag{3.16c}$$

Such equation must be solved to obtain ${}^{w}t_i$, $i = 1, 2, ..., N$. A system of $K \geq N$ linearly independent equations is needed. Such system can be formulated in matrix form as follows in Eq.(3.17).

$$\underbrace{\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ I & 0 & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ I & 0 & 0 & \dots & 0 \\ & \vdots & & \vdots & \end{bmatrix}}_{I_l} \underbrace{\begin{bmatrix} {}^{w}t_1 \\ {}^{w}t_2 \\ {}^{w}t_3 \\ \vdots \\ {}^{w}t_N \end{bmatrix}}_{v} = \underbrace{\begin{bmatrix} I & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & I \\ 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ & \vdots & & \vdots & \end{bmatrix}}_{I_r} \underbrace{\begin{bmatrix} {}^{w}t_1 \\ {}^{w}t_2 \\ {}^{w}t_3 \\ \vdots \\ {}^{w}t_N \end{bmatrix}}_{v} + \underbrace{\begin{bmatrix} {}^{w}R_1 \, {}^{1}t_2 \\ {}^{w}R_N \, {}^{N}t_1 \\ {}^{w}R_2 \, {}^{2}t_3 \\ {}^{w}R_3 \, {}^{3}t_1 \\ \vdots \end{bmatrix}}_{b} \tag{3.17}$$

which can be boiled down into

$$\underbrace{(I_l - I_r)}_{C} v = b. \tag{3.18}$$

Because there is noise, this equation will not be linearly independent and so there is no solution. To obtain an approximate solution for this overdetermined system, the LS solution is calculate through the pseudo inverse:

$$v = (C^T C)^{-1} C^T b. \tag{3.19}$$

v is then split up in 3×1 vectors which correspond to the best ${}^{w}t_i$ where $i = 1, 2, ..., N$ for this system.

3.4 Calibration Method

Using the techniques described in Section 3.2 and Section 3.3, a method can now be implemented to obtain the model of the calibration object. In this method, described in Fig. 3.4, we use a camera to capture images of the object. The procedure specified in B.3.1.A should be followed to maximize accuracy. At each captured frame the markers are found on the image. Using the, which are the points connected by the green line in Fig. 3.3, we can estimate the transformation of each marker in the reference frame of the camera, by solving a Perspective-n-Point (PnP) problem through the method described in [38]. These transformations are then paired with each other in order to get pairwise transformations, i.e. transformation between two markers. This process is repeated until there are enough transformations relating the markers, which will occur when, by using the pairwise transformations obtained, would be possible the chain them together in order to obtain the transformation from one marker to any other marker. Moreover, a large quantity of images should be acquired, so that many pairwise transformations will be obtained. This will prevent the effect of noise and of bad marker detections on the end result. Once the rotations are calculated, more pairwise transformations are obtained if needed. Once we have enough transformations, the translations are computed. This results in globally registered poses of the markers relative to a single reference frame as in Fig. 3.1b.

Figure 3.4: Marker Pose Estimation Flowchart.

It is important to note that due to the problem properties, it is possible to store information about the transformations in a compact manner, that remains in size no matter how many transformations were calculated. If we stored $I_0 - E$ from the rotations formulation in Eq.(3.9), and $I_l - I_r$ and b from the translations in (3.17), those matrices would grow with each pairwise transformation detected. This detections may be many in case we are running the method during a long period of time which may cause memory issues. To counteract this, we store at each time instant i for the rotations problem

$$B_i = (I_{0_i} - E_i)^T (I_{0_i} - E_i), \tag{3.20}$$

and for the translations problem

$$D_i = (I_{l_i} - I_{r_i})^T (I_{l_i} - I_{r_i}) \tag{3.21a}$$

$$y_i = (I_{l_i} - I_{r_i})^T b_i, \tag{3.21b}$$

Where I_{l_i},I_{r_i} and b_i are the elements of the problem specified in (3.17), obtained at a given instant.

Then due to mathematical properties it is possible to sum B_i,D_i and y_i from all instants L and obtain the exact same results as in Eq.(3.9) and (3.17):

$$B = \sum_{i=1}^{L} B_i = (I_0 - E)^T (I_0 - E). \tag{3.22}$$

The same reasoning is applied to D_i and y_i:

$$D = \sum_{i=1}^{L} D_i = (I_l - I_r)^T (I_l - I_r) \tag{3.23a}$$

$$y = \sum_{i=1}^{L} y_i = (I_l - I_r)^t b. \tag{3.23b}$$

Afterwards, to obtain the rotation we first obtain Z from Eq.(3.11). Instead of performing $\mathrm{SVD}(I_0 - E)$

eigendecomposition is performed $B = Q\Lambda Q^{-}1$ where Q holds an eigenvector in each column and Λ is a diagonal matrix holding the eigenvalues in descending order. We then extract the last 3 columns of Q which will correspond to Eq.(3.12).

Finally just like in Section 3.2 we split Z in 3×3 matrices and solve the orthogonal Procrustes problem to each of them as displayed in Eq.(3.13). To obtain the translations, instead of using the pseudo inverse formula in Eq.(3.19), we use:

$$v = D^{-1}y \tag{3.24}$$

v is then split up in 3×1 to obtain the translations of each marker relative to the world just like in the end of Section 3.3.

In the end of this process we obtain every marker's rotation and translation in the same reference frame. We can then represent the markers in 3D space as show in Fig. 3.5 where each marker is represented by a reference frame.

Figure 3.5: Obtained Marker Poses.

In order to facilitate posterior calculations, besides just having the markers transformation relative to the world, it is also important to have the corners of each of those markers in the same 3D reference frame, to later solve the PnP problem using those 3D corners.

Since the markers side $size$ is known beforehand, we know the positions of the corners relative to each marker as described in Eq.(3.25), where ${}^{M}t_{c_i}$ represents the position of corner i relative to marker M. This is considering that each marker has its reference frame as displayed in Fig. 3.6.

$$\begin{bmatrix} {}^{M}t_{c_1}{}^{T} \\ {}^{M}t_{c_2}{}^{T} \\ {}^{M}t_{c_3}{}^{T} \\ {}^{M}t_{c_4}{}^{T} \end{bmatrix} = \begin{bmatrix} -\frac{size}{2} & \frac{size}{2} & 0 \\ \frac{size}{2} & \frac{size}{2} & 0 \\ \frac{size}{2} & -\frac{size}{2} & 0 \\ -\frac{size}{2} & -\frac{size}{2} & 0 \end{bmatrix} \tag{3.25}$$

By applying the rigid transformation to every marker k and every marker corner i, we obtain every corners transformation relative to the same world referential as shown in Eq.(3.26).

Figure 3.6: Marker axis orientation.

$$^{w}t_{Cij} = {}^{w}R_{M_k}\,{}^{M}t_{c_i} + {}^{w}t_{M_k}, \tag{3.26}$$

With this, we have obtained the full calibration object model which in the next chapter will be used to obtain the camera poses. The model can be seen in Fig. 3.7b where each marker is represented by a reference system and their respective corners are connected with a red line.

(a) Calibration Object. **(b)** Model of Calibration Object.

Figure 3.7: Scenes.

4

Camera Network Calibration

Contents

4.1 Camera Pose Initialization 25

4.2 Camera Pose Fine Adjustments 26

In this chapter we go over the steps necessary to obtain camera poses that are very accurate. Firstly a rough estimation of the camera poses are acquired throught the method develop in this thesis. Afterwards we use Global Procrustes Analysis ICP (GPA-ICP) to refine this camera poses.

4.1 Camera Pose Initialization

Although getting to the transformations of the cameras in the same referential may look difficult at first, in actuality the problem stated in Section 3.1, can just as well be applied here. Having a set of cameras and a set of pairwise transformations between them, it is possible to obtain the transformation of every single camera with respect to one global frame (as shown in Fig. 4.1b)?. This means that equations and methods described in the previous chapter can also be used for the camera pose initialization. To get the transformations between cameras, they need to have in their view something in common, which in this case will be the calibration object from the previous chapter (Fig. 3.7). The current problem is described by image Fig. 4.1a, where frame B corresponds to the object frame, and C_1, C_2 and C_3 will be each camera, that observe the object and know its transformation relative to their referentials.

(a) Cameras observing calibration object. **(b)** Global Transformations.

Figure 4.1: Object Transformations.

To fit this into the previously defined problem formulation where we use pairwise transformation between objects, we compose pairs of transformations which are calculated as per Eq.(4.1b). In these equations the terms of the right side are the information retrieved by each camera regarding the calibration object.

$$ {}^{C_j}R_{C_i} = {}^{C_i}R_B \, {}^{C_j}R_B{}^T \tag{4.1a} $$

$$ {}^{C_j}t_{C_i} = {}^{C_j}R_B{}^T(-{}^{C_j}R_B{}^T \, {}^{C_j}t_B) + {}^{C_j}t_B \tag{4.1b} $$

4.1.1 Calibration Method

Details on how to move the calibration object and other conditions that improve the calibration accuracy are described in B.3.1.B. The calibration process described in flowchart 4.2, although similar to Section 3.4, now differs on its input. Previously, a single camera would visualize markers, where now every camera in the system captures an image at synchronized time instants. Then, each camera locates, if present, markers in its image, uses them to estimate the transformation between the calibration object and itself by solving a Perspective-n-Point (PnP) problem. The PnP problem uses the detected corner markers in each image, and its correspondence of the known model to estimate a transformation. These transformations are then used to generate pairwise transformations between pairs of cameras. Afterwards, if enough transformations were generated, that is, if the transformations from one camera to any other are defined, then the image acquisition halts and proceeds to estimate the rotations between each camera and the world. Subsequently, more images may be captured to get more pairwise transformations and when enough transformation have been obtained the translations are estimated.

Figure 4.2: Camera Pose Estimation Flowchart.

Having the rotations and the translations, we now know the transformation of each camera in world coordinates. As these transformations are not suffiently accurate, a fine adjustment must be applied.

4.2 Camera Pose Fine Adjustments

After the initial calibration from Section 4.1.1 is performed, sometimes the results may not be accurate enough, this can be due to the following reasons:

- Incorrect model of the calibration object.
- Incorrect camera intrinsic parameters.
- Unsynchronized image acquisition between cameras.
- Fast moving calibration object.
- Extreme lighting conditions

In order to fix the misalignment that might exist between cameras, a fine calibration step is executed. This step consists in using the point clouds, i.e. the depth data originated from each camera, to perform the alignment. In Section 2.2 some of the algorithms capable of this task were described. Of those, Iterative Closest Point (ICP) [32], Joint Registration of Multiple Point Clouds (JR-MPC) [35] and GPA-ICP [7] were tested. It was understood that because ICP registers point clouds by incrementally registering them onto the previously registered ones, it will generate results that differ depending on the order chosen to register the clouds and it will also generate error with each subsequent registration. JR-MPC was also tested, but it was concluded that for it to yield accurate results there must be a lot of overlap between all the point clouds we wish to register. In the end, the method that yielded a better result was GPA-ICP. This algorithm combines the Generalized Procrustes Analysis (GPA) with the ICP in order to achieve a global registration method for multiple point clouds. Because this method registers multiple point clouds simultaneously, it will distribute the errors evenly, as opposed to sequential registration approaches which registrate point clouds in pairs leading to the accumulation of error with each individual registration.

4.2.1 General Procrustes Analysis

Consider the Orthogonal Procrustes Problem Eq.(3.13), where the solution yields the orthogonal matrix R, that best aligns the rows of point sets A and B, which contain p points with k dimensions.

This can be extended to also calculate a scale and a translation between sets, called the Extended Orthogonal Procrustes Analysis (EOP), in which the function to minimize is represented in Eq.(4.2), where A and B are two $p \times k$ matrices that we wish to fit, t is the $k \times 1$ translation vector, R is the $k \times k$ rotation matrix [39], 1_p is a $p \times 1$ vector of ones and c is the scale factor.

$$\begin{aligned} \underset{R,c,t}{\arg\min} \quad & ||cAR + 1_p t^T - B||_F^2 \\ \text{subject to} \quad & R^T R = I \end{aligned} \tag{4.2}$$

This can be further extended into a Weighted Extended Orthogonal Procrustes Analysis (WEOP) where we minimize Eq.(4.3), where weights are added, for points and for dimensions, W_P is a $p \times p$ matrix, and W_K is a $k \times k$ diagonal matrix. If $W_K = I$ a closed form solution solves the unknowns [39], otherwise, iterative methods have to be used.

$$\begin{aligned} \underset{R,c,t}{\arg\min} \quad & \operatorname{tr}\left(\left(cAR + 1_p t^T - B\right)^T W_P \left(cAR + 1_p t^T - B\right) W_K\right) \\ \text{subject to} \quad & R^T R = I \end{aligned} \tag{4.3}$$

One of the solutions can be given by the Generalized Orthogonal Procrustes Analysis which provides the least squares solution, for more than 2 point sets Eq.(4.4).

$$\begin{aligned} \underset{R_i, c_i, t_i}{\arg\min} \quad & \operatorname{tr}\left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\left(c_i X_i R_i + 1_p t_i^T\right) - \left(c_j X_j R_j + 1_p t_j^T\right) \right)^T \left(\left(c_i X_i R_i + 1_p t_i^T\right) - \left(c_j X_j R_j + 1_p t_j^T\right) \right) \right) \\ \text{subject to} \quad & R^T R = I \end{aligned} \tag{4.4}$$

In this problem, $X_i, \quad i = 1, 2..N$ are sets of p points in k dimensions, that have correspondences across all sets. This can be converted onto the formulation in Eq.(4.5), where to obtain the transformations the geometrical centroid K is initialized, a loop is run where at each iteration a solution is calculated for each point set X_i with respect to that centroid K using WEOP, the obtained rotation, translation and scale are applied to it and the geometrical centroid K is updated Eq.(4.6). This process repeats itself until global convergence is achieved, i.e. when the position of the centroid stabilizes.

$$\begin{aligned} \hat{R}_i, \hat{t}_i = \underset{R_i, t_i}{\arg\min} \quad & \sum_{i=1}^{N} \left\| \hat{X}_i - K \right\|^2 = \sum_{i=1}^{N} \operatorname{tr}\left(\hat{X}_i - K\right)^{\mathrm{T}} \left(\hat{X}_i - K\right), \\ \text{subject to} \quad & R_i \in SO(3), \\ & i = 1, ..., N \end{aligned} \tag{4.5}$$

where $\hat{X}_i = c_i X_i R_i + 1 t_i^t$ as described in [7].

$$K = \frac{1}{N} \sum_{i=1}^{N} \hat{X}_i \tag{4.6}$$

One final addition to General Procrustes that is very useful in real applications is the possibility for partial correspondences. Sometimes points in one of the point sets, may have correspondences in only some of the other sets. This addition is done by using an M_i matrix associated to each point set which is a diagonal binary $p \times p$ matrix, that by setting a diagonal element to 0 or 1 will define if that point indeed exists in that set or not. The equation with this addition is described in Eq.(4.7), where the centroid K must now take into account which are the active points in each set Eq.(4.8)

$$\begin{aligned} \hat{R}_i, \hat{t}_i = \underset{R_i, t_i}{\arg\min} \quad & \sum_{i=1}^{m} \operatorname{tr}\left(\left(\hat{X}_i - K\right)^T M_i \left(\hat{X}_i^{\,p} - K\right) \right) \\ \text{subject to} \quad & R_i \in SO(3), \\ & i = 1, ..., N \end{aligned} \tag{4.7}$$

$$K = \left(\sum_{i=1}^{m} M_i \right)^{-1} \left(\sum_{i=1}^{m} M_i \left(c_i X_i R_i + 1_p t_i^T\right) \right) \tag{4.8}$$

4.2.2 Algorithm for multiple point cloud registration

In order to register multiple point clouds, the authors in [7] use the GPA previously described, where the match matrix M_i is defined through a specific method. The only points in each set that are considered are the ones that are mutually the nearest neighbor of each other across all sets. This matching is done in two steps. First for each point in each set, their closest neighbor is found by calculating the Euclidean norm in every other point set as shown in Fig. 4.3a where the dotted line represents a one way neighbor relationship, while a full line represents a mutually nearest neighbor. Then those mutually neighborships are chained together into groups. A group of neighborships is only considered valid if there is a maximum of one point per set as shown in Fig. 4.3b. The matching matrix M_i for each point set i is updated accordingly, and so points may have different matches from iteration to iteration. Having this new M_i we can initialize or update the geometrical centroid K by having each group of mutually neighboring points generate their centroid as per Eq.(4.8). Because of this, points from those same groups which are matched to the same point, will all be driven towards the same position.

(a) Nearest Neighbor vs Mutual Nearest Neighbor. **(b)** Independent Group Sets.

Figure 4.3: Point matching. Images taken from [7].

This matching method prevents points that are very far away from each other from compromising the results of GPA, leading to better point cloud matching. The algorithm results in the following steps:

1. Find the mutually nearest neighbors among point sets.

2. Define a centroid for each independent mutually nearest neighbor set.

3. Estimate transformations for each point cloud using Eq.(4.7).

4. Transform each point cloud using the estimated transformations.

5. Go back to step 1, until global convergence or the maximum number of iterations is reached.

Several experiments are performed, and the results are interpreted in Chapter 5. In general the algorithm proposed in [7] yields good results as long as the initial point cloud poses are close enough to the real camera poses as shown in Fig. 4.4.

(a) Robot Arm on Table Before Algorithm.

(b) Robot Arm on Table After Algorithm.

Figure 4.4: GPA-ICP Before vs After

5

Experimental Evaluation

Contents

5.1 Calibration Object Experiments 32

5.2 Camera Poses Experiments 37

5.3 3D Data Fusion 42

5.4 Limitations 44

In this chapter we go over the results in each step in the process to get the camera poses of the network. We start by describing and analysing the results regarding the obtained model for the calibration object. Next, experiments on obtaining the initial camera poses are carried out, and its outcomes discussed. Finally we verify the influence of the Global Procrustes Analysis ICP (GPA-ICP) algorithm on the previously obtained poses, and conclude on how to use it in order to maximize its accuracy.

5.1 Calibration Object Experiments

5.1.1 Description of the Calibration Object

We use a calibration object a set of ArUco markers, a well known fiducial marker with existing libraries to detect and locate them, namely OpenCV[1]. The markers are placed in every side of the calibration object shown in Fig. 5.1a so that multiple cameras, positioned at different standpoints, can find the calibration object using the markers detected in their view. Furthermore each of the markers must be unique in order to distinguish them from each other. Their sizes must also be the same (each marker in Fig. 5.1 has $8.75\ cm$ of side length). The marker IDs for the calibration object used can be found in Fig. 5.1b.

(a) Real calibration object.

(b) Unfolded calibration objectwith IDs.

Figure 5.1: Calibration Object

We use the object in Fig. 5.1a across all the camera network calibration tests in this chapter. Other

[1] OpenCV ArUco Detection: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html

object structures are also modeled but are not used after that.

5.1.2 Calibration Object Evaluation

The calibration object is placed near the camera as in Fig. B.3, and the procedure in Section 3.4 is then followed. The resulting model is shown in Fig. 5.2b where the red line represents the border of every markers, and each reference frame represents their position and orientation, which we can see is similar to the actual object in Fig. 5.2a.

(a) Calibration Object. **(b)** Model of calibration Object.

Figure 5.2: Calibration Object vs Model comparison

The obtained model can be compared to the real calibration object, by measuring distances between corners of different markers. So to understand the correctness of the model, we measure the diagonal of each face of the calibration object, that is from the bottom left corner of the marker on the bottom, to the top right corner of the marker on the top, using a ruler with ± 0.0005mm uncertainty. Table 5.1 displays the measurements and how they compare. The values obtained from the model do not deviate more than 3mm from the ground truth model.

Afterwards, to measure how well a camera can detect the calibration objection using the built model, we measure the error on the captured calibration objection model, we measure the self reprojection error $\bar{E}_r$ in Eq.(5.1) at each frame.

Table 5.1: Diagonal Measurements.

Side Diagonal [m]	1	2	3	4
Ground Truth	0.300	0.296	0.290	0.294
Model	0.303	0.298	0.292	0.296
Absolute Error	0.003	0.002	0.002	0.002
Relative Error	1.06%	0.64%	0.75%	0.67%

$$\bar{E}_r = \frac{\sum_{i=1}^{N} ||p_i - \hat{p}_i||_F^2}{N}, \tag{5.1}$$

In this equation, p_i correspond to the pixel coordinates of a point from the detected object, and $\hat{p}_i$ are the pixel coordinates of the corresponding point on the obtained model. This pixel coordinates can be obtained through the method explained in appendix A. N is the total number of points captured. This metric is used to compare the corners observed in the image, with the points obtained from reprojecting the model in each individual frame. The corners this metric uses can be visualized in Fig. 5.5a where the corners detected in the image are in green and the ones reprojected from the model are in red.

In Fig. 5.3 we see how it evolves over time. From this graphic we conclude that the error is stable, which means that most of the time, the detected corners and the reprojected corners stay around the same pixel distance. The gap in the graphic refers to a period where no marker was detected, and consequently, no error was calculated.

Figure 5.3: Average Reprojection Error in one of the experiments.

By plotting the histogram for this data, in Fig. 5.4, we see that the error distribution resembles a Gaussian distribution with the parameters in Table 5.2, where we see that on average, each reprojected corner is 0.98 pixels apart from the corresponding detected corners, and from the standard deviation we can interpret that 99.7% (3σ) of the frames have a detection error between 0.05 and 1.91px.

Some outliers can also be distinguished on the histogram. These are the result of frames where corners of the detected markers are incorrectly estimated due to blur and/or to an unfavorable viewing angle. Specifically, frame 619 has the worst outlier, Fig. 5.5b shows a zoom of that frame where we

Figure 5.4: Average Reprojection Error in one of the experiments.

Table 5.2: Reprojection Error Statistics [px].

	Mean μ	Standard Deviation σ
Reprojection Error [px]	0.98	0.31

see those two situations occurring. Because the markers on the right side have a very steep angle, the detected corners are somewhat deviated from the correct space, which then leads to a bad estimation of the calibration object transformation, resulting in reprojected corners that are on average 2.56 pixels apart from the corresponding detections. When compared to the detection on frame 53, for example, with an error of 1.0 pixels, we can visualize how the effects described above influence the detection comparing to when they are absent in Fig. 5.5a

5.1.3 Alternative Calibration Objects

To test the robustness of the implementation, we modeled two other calibration objects. The first one is a cube shaped box seen in Fig. 5.6a with one marker per side. Because it has a simple geometry with easily identifiable markers and it is possible to detect two or three markers at a time, it yielded an accurate model (Fig. 5.6).

The second calibration object to obtain is a whole area with markers spread all around it. Markers are spread across the room as in Fig. 5.7a, so instead of moving the calibration object in front of the camera, we move the camera around the scene in order to capture transformations for all the existing markers. Since the used camera has a low resolution (640×480 pixels) it has to be relatively close to the markers to detect them and has so, because the markers are more sparsely distributed, there are

(a) Detections and reprojections on frame 53.

(b) Detections and reprojections on frame 619.

Figure 5.5: Detections and reprojections comparison.

(a) Calibration Object.

(b) Model of calibration Object.

Figure 5.6: Cube Calibration Object.

fewer pairwise connections. For example the marker on the chair on the far right, can only be seen at the same time as two other markers (the one on the right monitor and on the desk side). This yields a less connected pose graph of the markers that will result in an inaccurate estimated model. This can be confirmed in Fig. 5.7b where, although most markers are around the correct position, the one on the right side chair is incorrectly estimated.

(a) Calibration Object. **(b)** Model of calibration Object.

Figure 5.7: Scene Calibration Object.

5.2 Camera Poses Experiments

5.2.1 Qualitative Experiments on Camera Poses

To aforementioned calibration object from Fig. 5.6 is used in our system to obtain the camera poses. The correct way to use it is explained in the Appendix B.3.1.B. After running the method in two different scenarios, we can see qualitatively in images 5.8 and 5.9 how the modeled camera poses on the right, match the real camera poses on the left.

(a) Real Camera Configurations. **(b)** Obtained Camera Transformations.

Figure 5.8: Camera Poses in Test 1.

From these images we comprehend that the method produces poses similar to the real ones, but this results are greatly dependent on how accurately we know the calibration object and how close we move

(a) Obtained Camera Transformations.

(b) Obtained Camera Transformations.

Figure 5.9: Camera Poses in Test 2.

that object to the cameras, as the further away it is, the worse the results will be. A more extensive error analysis will be performed in 5.2.3.

5.2.2 Fine Tuning The results

To motivate the importance of having good pose initializations for the correct convergence of GPA-ICP, the algorithm was run with and without this camera pose initializations. By inspection of Figures 5.10 and 5.11, we can see how GPA-ICP behaves without and with an initialization respectively. Just by observation we can see that having a good initialization induces the algorithm converge to the correct poses.

(a) Before.

(b) After.

Figure 5.10: Fine tuning when without initialization.

(a) Before. **(b)** After.

Figure 5.11: Fine tuning using initialization from test 1.

5.2.3 Error Analysis of the Camera Poses

To understand how much of an impact the initialization and GPA-ICP steps described in Chapter 4 have in generating poses that resemble the real ones, a new camera setup where all the cameras are pointing to a zone was mounted as in Fig. 5.12.

Figure 5.12: Test 3: Camera Poses Pointing at an robot arm.

We first measured the distances between every camera pair using a metric tape with $\pm 0.0005\ mm$ uncertainty, this metrics were considered as the ground truth. We then ran 3 independent pose initializations, from which the resulting poses will vary depending on factors such as the way we moved the calibration object in each initialization or on the amount of pairwise transformations captured. We then also captured a point clouds from the scene in Fig. 5.12 where each camera will have in its respective point cloud a part of the robot arm. The camera poses before and after the algorithm was run were then compared to the ground truth, leading to table 5.3. C_i-C_j corresponds to metrics taken between camera i and camera j. This table displays the absolute and relative errors calculated for each initialization, before and after GPA-ICP was applied.

From it we can understand that with just the initialization, the obtained poses are erroneous and dependent on how the calibration object was moved, since for one initialization a camera pair may

have very little error, while in another the same pair will be very inaccurately estimated. For example, distances between C1-C2 for initialization 1 give out the biggest error since it deviates 10.8cm from the ground truth. At the same time the same camera pair has almost no error for the other initializations with an absolute error of 1.2cm and $4.6cm$. The Average column, shows us the average error for that rown. From it we can comprehend that the average error for each initializations is very similar varying between 5.15% and 6.77%.

By performing the GPA-ICP algorithm for each initialization, we obtain better poses expressed by the overall drop in error after this step is performed. This step is particularly important to refine poses that generate camera distances with a relative error above 10% since we see all of those values drop below 5%, for example in C2-C3 of initialization 3, this step turns a distance error of 10.8cm to 0.06cm achieving sub centimeter accuracy. However it seems this step improves the overall poses given from the initializations, at the cost of slightly increasing the error in one of the distances. We can see this occur for distances measured in C2-C3, C1-C3 and C1-C3 for initial poses 1, 2 and 3 respectively. These three pairs contain camera 3. By analysing Fig. 5.12, we understand that camera 3 generates the point cloud that is most different from the others, since it mostly observes the side of the robot, while the others mostly observe the front. This information leads us to conclude that algorithm, favors point clouds with more similarities, just as it is described in Section 4.2.2, while not acting in the poses of point clouds that don't have many similarities, i.e. overlap with the others.

From this we understand that the camera setup, and the part of the scene that each camera sees, will affect the results of the algorithm. To explore this in more detail, the camera setup remained the

Table 5.3: Camera Pose Errors for 3 initializations before and after GPA-ICP.

	Poses Step	C1-C2	C1-C3	C1-C4	C2-C3	C2-C4	C3-C4	Average
Absolute Error [m]	1 before	0.108	0.035	0.092	0.017	0.013	0.033	0.050
	1 after	0.023	0.011	0.019	0.036	0.006	0.014	0.018
	2 before	0.012	0.016	0.052	0.084	0.062	0.056	0.047
	2 after	0.003	0.025	0.016	0.006	0.015	0.041	0.018
	3 before	0.046	0.009	0.067	0.108	0.051	0.071	0.058
	3 after	0.008	0.022	0.012	0.005	0.003	0.042	0.015
Relative Error [%]	1 before	19.69	8.03	6.92	2.16	1.47	2.37	6.77
	1 after	4.18	2.58	1.43	4.55	0.61	1.03	2.40
	2 before	2.11	3.60	3.89	10.51	6.78	4.02	5.15
	2 after	0.53	5.73	1.16	0.78	1.68	2.96	2.14
	3 before	8.31	2.04	4.98	13.50	5.57	5.09	6.58
	3 after	1.40	5.20	0.90	0.59	0.30	3.03	1.90

same but the scene changed. In scene 2 the chair and robot arm are replaced with a person facing the cameras and in scene 3 it is tested how the algorithm deals when there most there is overlap between all camera views. Initialization 1 was chosen as a starting point for all of this tests because it was the one most favorable to scene 1 in Fig. 5.12 as it made the relative error decrease the most. Table 5.4 displays the distance errors between pairs of cameras for different observed scenes in order to assess their effect in the performance of GPA-ICP.

Table 5.4: Effect of different scenes in GPA-ICP performance.

		C1-C2	C1-C3	C1-C4	C2-C3	C2-C4	C3-C4	Average
Absolute Error [m]	Before	0.108	0.035	0.092	0.017	0.013	0.033	0.050
	After (Scene 1)	0.023	0.011	0.019	0.036	0.006	0.014	0.018
	After (Scene 2)	0.363	0.112	0.249	0.036	0.003	0.010	0.129
	After (Scene 3)	0.180	0.007	0.158	0.095	0.013	0.129	0.097
Relative Error [%]	Before	19.69	8.03	6.92	2.16	1.47	2.37	6.77
	After (Scene 1)	4.18	2.58	1.43	4.55	0.61	1.03	2.40
	After (Scene 2)	65.91	25.96	18.67	4.55	0.31	0.73	19.36
	After (Scene 3)	32.76	1.60	11.83	11.81	1.38	9.29	11.45

By looking at the average relative errors we can see that although the algorithm functions well for scene 1, it mishandles the other two. In scene 2 we can see that the relative distance errors for camera 1 greatly increase. By looking at the individual point clouds that each camera produces we understand that camera 1, is the one with least points in common with the rest, resulting in it transformation to remain unchanged when the algorithm is applied.

For scene 3 all distance errors increase but one, However by looking at Fig. 5.13 we see that the algorithm aligns the points clouds correctly. From this we understand, that the algorithm found another set of camera poses that does not correspond to the real poses,and yet it still correctly aligns the point clouds. This ocurred because the point clouds used do not have a large quantity of points in them.

(a) Scene 3 Before. **(b)** Scene 3 After.

Figure 5.13: Scene 3 Before vs After GPA-ICP.

From all the tests performed we conclude that, the method developed in this thesis to generate initializations manages to generate camera poses where cameras position are miscalculated by roughly

5 centimeters on average as per Table 5.3. Moreover its results depend on how the calibration object is moved around the scene during the calibration process. Regarding the GPA-ICP, we come to the understanding it that it only acts on point clouds that have some overlap between cameras, and so, for it to return accurate results, there must be a fair amount of overlap between pairs of cameras.

5.3 3D Data Fusion

Having the camera poses directly from our proposed method, and after GPA-ICP is performed it is possible to compare their accuracy by merging the point clouds from every camera. By cropping from the point clouds, points that are too far away (> 2 meters), and by applying the camera pose initializations from Test 2 shown in Fig. 5.9b, we obtain the point clouds from the left side of images of 5.14 and 5.15.

(a) Calibration Object on Table Before.

(b) Calibration Object on Table After.

(c) Person Before.

(d) Person After.

Figure 5.14: Before vs After GPA-ICP Part 1.

(a) Calibration Object on Box Before.

(b) Calibration Object on Box After.

(c) Robot Arm on Table Before.

(d) Robot Arm on Table After.

Figure 5.15: Before vs After GPA-ICP Part 2.

From this images we apprehend that the poses obtained in test 2 initializations cause some misalignment between point clouds. Moreover, the calculated poses have mostly an error in their translation, as rotation correctly matches, this is distinctly seen in Fig. 5.14a. This error in translation may exist because of inaccurate camera intrinsic parameters which may affect how distances are perceived. Another factor is the pixel discretization error, which occurs when we attempt to project a certain point in 3D space into an image pixel. The further away the 3D points are from the camera, the more uncertanty will exist on their position, as a pixel corresponds to a volume in 3D space from which point contained in it are projected onto that pixel.

After using GPA-ICP the clouds from the right side of Fig. 5.14 and Fig. 5.15 are obtained, which just by observation we can conclude that they are better registered, and so the camera poses that generate this set of registered clouds will also be more accurate.

5.4 Limitations

The developed pose initialization method has no outstanding limitations aside from depending an RGB image to determine distances, which leads to not very accurate transformations. This is because of possible inaccurate marker detections, incorrect intrinsic parameters or simply a low image resolution. However, the designed camera network system described in Appendix B where the full calibration was tested has some limitations which are described in detail in Section B.4.

6

Application

Contents

6.1 Description 46
6.2 Results 47

6.1 Description

Nowadays, due to better healthcare and life conditions, the overall life expectancy has increased [40]. In turn, the elder population and the disabled population have grown. This leads to a high demand for caregivers, which at the current rate will not be able to be supplied if the approach that the public services take to caring stays as it is [41]. One way to ease the work of the existing caregivers, allowing them to perform more differentiated tasks and give aid to more people, is to automate some of the more simple tasks. Some of this activities are contained within the Activities of Daily Living (ADL) set, which includes things such as bathing, feeding, toileting and ambulation [42]. With the help of a robotic arm or a mobile robot, some of this activities can be assisted, giving more autonomy to this endangered classes.

One of the main motivations on this work is to extend Feedbot [5], a robotic arm system that aids a person in the process of eating his or her meal. The current system, displayed in Fig. 6.1 includes two elements. One of them is a robot arm with a spoon as an end-effector used to pick up food, and move it towards the mouth of the person. The second is an RGB-D camera that is used to track the pose of the user's mouth. The work developed in this thesis can extend this system by using multiple RGB-D cameras to generate more complete 3D reconstructions in real-time, which can be used to better detect the user and even caregivers. The other way it can extend it is by using multiple RGB-D cameras to first generate a more complete model of the users head so that it is easier to track him, and his mouth over time. The extension was tested in the Feedbot system because no physical changes have to be done, the only needed change is to replace the current face model that the system uses for the one generated by us.

Figure 6.1: Feedbot Feeding System.

There are other readily available robot arms that automate the feeding process, such as iEAT [43], bestic [44] and obi [45]. The issue with this systems is that their end-effector follows a previously defined trajectory, that does not adapt to the user. Besides Feedbot, the other robot arm currently being developed that adapts to the user (and also to the food on the plate) is described in [46–48]. Moreover, their system makes use of an RGB-D camera to track the mouth of the user, and an RGB camera that sees the food in the plate and detects what it is, in order to pick it up in a specific manner.

In summary, one of the main challenges of this type of application is to identify where a person's mouth is located. Feedbot solves this by using a 3D model of the person's whole face (or head), and then tracking it over time using Discriminative Optimization (DO) on the point cloud information retrieved from a RGB-D camera.

The DO method developed in [49] is used for tracking using real-time point cloud matching. It learns in which directions it should search and then moves one point cloud a slight amount to better align to the other. This process is repeated for fixed number of iterations at the end of which it is expected for transformation that best matches the point clouds to be known. When a bad initialization is given i.e. the point clouds to match have a very different rotation and transformation, its performance is far greater than Iterative Closest Point (ICP) which would most likely converge onto a local minimum. This occurs because this method learns in which directions to search for the global minimum.

6.2 Results

The complete 3D face model from Fig. 6.2 was obtained using the setup of test 2 (See Fig. 5.8) by placing a user in front of the cameras, and cropping the points relative to his face. To verify that it improves the DO results, we then used DO to match this model with a partial 3D face. The results are shown in Fig. 6.3, where the data we want to fit the model to is in blue and the model itself is in green. In every case, the model converges to the partial data.

To prove that this 3D full face model improves convergence when compared to a less complete face model, we ran more tests, where the used model is only a partial 3D face. In Fig. 6.4, we can see the results for both situations. In Fig. 6.4a due to bad initialization and lack of similarity with the model, DO does not converge correctly, while in Fig. 6.4b it does.

Afterwards, the full 3D face model is used in Feedbot to check how well it helps the system find the face of the user face in real time. It can be seen in Fig. 6.5 that the face detection software based on DO correctly finds the face of the user face and using facial landmarks it finds the mouth, represented by a red sphere. The robot then uses this position to move the food filled spoon towards the users' mouth.

(a) Face Side View. **(b)** Face Top View. **(c)** 3D Model Colored Face.

Figure 6.2: 3D Face Model Obtained.

(a) Model alignment on point cloud with partial noisy face data. **(b)** Model alignment on point cloud with partial noisy face data and outliers.

Figure 6.3: Discriminative Optimization Results.

(a) DO result is misaligned when partial face model is used. **(b)** DO result is aligned when full face model is used.

Figure 6.4: Partial face detection with DO.

Figure 6.5: Computer vizualisation of the Feedbot system.

7

Conclusions & Future Work

Contents

7.1 Conclusions 51

7.2 Future Work 51

7.1 Conclusions

In this work we created a methodology to easily calibrate any network of RGB-D cameras. This system can be used to obtain a 3D Reconstruction of the surround space and specific objects. This methodology comprises 2 steps. The first one consists in an initialization of the camera poses, while the second refines this results using a point cloud matching algorithm. This combination led to a very accurate camera registration.

We use existing metrics and created new ones to measure the accuracy of the results for every step of the progress, and reached the conclusion that the method works, is quick, accurate and it is easily expandable to include more, and different kinds of cameras.

An application setting was tested, Feedbot, a feeding a robot arm, to aid people with upper body mobility impairments to eat their meals. The applicability of the method was proved by the well-succeeded tracking of a face using our full 3D face model as a reference, thus allowing the robot to find the person's mouth.

7.2 Future Work

In the future we wish to combat some of the current system limitations, so as to enable easier scalability. This means making a more distributed system where each camera sends less information through the network and making it more robust to possible camera desynchronization issues. Morerover, we wish to use other types of objects for the initial calibrations, that are not fiducial markers, to further extend the usability of the method. Finally also intend to make the calibration method more user-friendly, by making it automatic. Finally we will add to the system the ability to track moving RGB-D cameras, obtaining merged moving point clouds as a result.

Bibliography

[1] "Allegory of the cave." [Online]. Available: https://web.stanford.edu/class/ihum40/cave.pdf

[2] "Kinect for Windows." [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect

[3] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418. [Online]. Available: https://ieeexplore.ieee.org/document/6619156

[4] C. Chen, J. Favre, G. Kurillo, T. P. Andriacchi, R. Bajcsy, and R. Chellappa, "Camera networks for healthcare, teleimmersion, and surveillance," *Computer*, vol. 47, no. 5, pp. 26–36, May 2014. [Online]. Available: https://doi.org/10.1109/MC.2014.112

[5] A. Candeias, T. Rhodes, M. Marques, J. P. ao Costeira, and M. Veloso, "Vision augmented robot feeding," in *The European Conference on Computer Vision (ECCV) Workshops*, September 2018. [Online]. Available: http://openaccess.thecvf.com/content_eccv_2018_workshops/w31/html/Candeias_Vision_Augmented_Robot_Feeding_ECCVW_2018_paper.html

[6] "Feedbot: A Vision Augmented Feeding Robot." [Online]. Available: https://drive.google.com/file/d/1rCMi43s6y0wthHD6OuU5Nu8QXi-8qdTE/view

[7] R. Toldo, A. Beinat, and F. Crosilla, "Global registration of multiple point clouds embedding the generalized procrustes analysis into an icp framework," in *3DPVT 2010 Conference*, vol. 2, 2010. [Online]. Available: https://www.researchgate.net/publication/228959196

[8] "Vicon — award winning motion capture systems." [Online]. Available: https://www.vicon.com/hardware/cameras/

[9] "The Captury — Markerless Motion Capture technology." [Online]. Available: https://thecaptury.com/

[10] "Qualisys — motion capture systems." [Online]. Available: https://www.qualisys.com/hardware/

[11] J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips, “Extrinsic calibration of camera networks using a sphere,” *Sensors*, vol. 15, no. 8, pp. 18 985–19 005, 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4570356/

[12] M. Ruan and D. Huber, “Calibration of 3d sensors using a spherical target,” in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 187–193. [Online]. Available: https://doi.org/10.1109/3DV.2014.100

[13] P.-C. Su, J. Shen, W. Xu, S.-C. Cheung, and Y. Luo, “A fast and robust extrinsic calibration for rgb-d camera networks,” *Sensors*, vol. 18, no. 1, p. 235, 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29342968

[14] J. Shen, W. Xu, Y. Luo, P.-C. Su, and S.-c. S. Cheung, “Extrinsic calibration for wide-baseline rgb-d camera network,” in *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2014, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/6958798

[15] G. Kurillo, Zeyu Li, and R. Bajcsy, “Wide-area external multi-camera calibration using vision graphs and virtual calibration object,” in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, Sep. 2008, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/4635695

[16] T. Svoboda, D. Martinec, and T. Pajdla, “A convenient multicamera self-calibration for virtual environments,” *Presence: Teleoper. Virtual Environ.*, vol. 14, no. 4, pp. 407–422, Aug. 2005. [Online]. Available: http://dx.doi.org/10.1162/105474605774785325

[17] X. Chen, J. Davis, and P. Slusallek, “Wide area camera calibration using virtual calibration objects,” in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, June 2000, pp. 520–527 vol.2. [Online]. Available: https://doi.org/10.1109/CVPR.2000.854901

[18] E. Auvinet, J. Meunier, and F. Multon, “Multiple depth cameras calibration and body volume reconstruction for gait analysis,” in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, July 2012, pp. 478–483. [Online]. Available: https://ieeexplore.ieee.org/document/6310598

[19] M. Munaro, F. Basso, and E. Menegatti, “Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks,” *Robotics and Autonomous Systems*, vol. 75, pp. 525–538, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015002304

[20] J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips, “Extrinsic calibration of camera networks based on pedestrians,” *Sensors*, vol. 16, no. 5, p. 654, 5 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/5/654

[21] C. T. Aslan, K. Bernardin, and R. Stiefelhagen, “Automatic Calibration of Camera Networks based on Local Motion Features,” in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*. Marseille, France: Andrea Cavallaro and Hamid Aghajan, Oct. 2008. [Online]. Available: https://hal.inria.fr/inria-00326791

[22] K. Desai, B. Prabhakaran, and S. Raghuraman, “Skeleton-based continuous extrinsic calibration of multiple RGB-D kinect cameras,” *Proceedings of the 9th ACM Multimedia Systems Conference, MMSys 2018*, vol. 18, pp. 250–257, 2018. [Online]. Available: https://doi.org/10.1145/3204949.3204969

[23] F. Basso, R. Levorato, M. Munaro, and E. Menegatti, *A Distributed Calibration Algorithm for Color and Range Camera Networks*. Springer, 02 2016, vol. 625, pp. 413–436. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-26054-9_16

[24] P. Dai, X. Wang, Y. Tan, P. Zhang, and X. Li, “Adaptive Recursive Optimized Extrinsic Self-calibration in Distributed Visual Sensor Networks,” in *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/I2MTC.2016.7520360

[25] M. Kowalski, J. Naruniec, and M. Daniluk, “Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors,” in *2015 International Conference on 3D Vision*, Oct 2015, pp. 318–325. [Online]. Available: http://dx.doi.org/10.1109/3DV.2015.43

[26] E. Ujkani, J. Dybedal, A. Aalerud, K. B. Kaldestad, and G. Hovland, “Visual marker guided point cloud registration in a large multi-sensor industrial robot cell,” in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, July 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/MESA.2018.8449195

[27] A. Aalerud, J. Dybedal, and G. Hovland, “Automatic calibration of an industrial rgb-d camera network using retroreflective fiducial markers,” *Sensors*, vol. 19, no. 7, p. 1561, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1561

[28] H. Afzal, D. Aouada, D. Font, B. Mirbach, and B. Ottersten, “RGB-D Multi-view System Calibration for Full 3D Scene Reconstruction,” in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 2459–2464. [Online]. Available: https://ieeexplore.ieee.org/document/6977138

[29] Q. V. Le and A. Y. Ng, “Joint Calibration of Multiple Sensors,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3651–3658. [Online]. Available: http://www.robotics.stanford.edu/~ang/papers/iros09-JointCalibrationMultipleSensors.pdf

[30] D. S. Alexiadis, D. Zarpalas, and P. Daras, “Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras,” *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 339–358, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6359953

[31] CMU, “CMU Panoptic Dataset,” 2015. [Online]. Available: http://domedb.perception.cs.cmu.edu/dataset.html

[32] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992. [Online]. Available: https://ieeexplore.ieee.org/document/121791

[33] P. Date, F. Leica, C. Register, H. D. S. Software, P. Management, L. Setup, B. Workflow, B. Configuration, G. Settings, L. Settings, and G. R. Information, “Leica Geosystems,” 2018. [Online]. Available: www.leica-geosystems.com

[34] U. Castellani, A. Fusiello, and V. Murino, “Registration of multiple acoustic range views for underwater scene reconstruction,” *Computer Vision and Image Understanding*, vol. 87, no. 1-3, pp. 78–89, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314202909847

[35] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis, “A Generative Model for the Joint Registration of Multiple Point Sets,” in *European Conference on Computer Vision*, vol. 8695 LNCS. Springer, 2014, pp. 109–122. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01019661v3

[36] B. Eckart, K. Kim, and J. Kautz, “Fast and Accurate Point Cloud Registration using Trees of Gaussian Mixtures,” *arXiv preprint arXiv:1807.02587*, 2018. [Online]. Available: http://arxiv.org/abs/1807.02587

[37] D. W. Eggert, A. Lorusso, and R. B. Fisher, “Estimating 3-d rigid body transformations: A comparison of four major algorithms,” *Machine vision and applications*, vol. 9, no. 5-6, pp. 272–290, 1997. [Online]. Available: http://dx.doi.org/10.1007/s001380050048

[38] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[39] D. Akca, “Generalized procrustes analysis and its applications in photogrammetry,” ETH Zurich, Tech. Rep., 2003. [Online]. Available: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/147853/1/eth-26937-01.pdf

[40] UN DESA, “World Population Prospects 2019: Highlights — Multimedia Library - United Nations Department of Economic and Social Affairs,” 2019. [Online]. Available: https://www.un.org/development/desa/publications/world-population-prospects-2019-highlights.html

[41] H. Gautun and C. Bratt, “Caring too much? Lack of public services to older people reduces attendance at work among their children,” *European Journal of Ageing*, vol. 14, no. 2, pp. 155–166, 2017. [Online]. Available: https://doi.org/10.1007/s10433-016-0403-2

[42] M. E. Mlinac and M. C. Feng, “Assessment of Activities of Daily Living, Self-Care, and Independence,” *Archives of Clinical Neuropsychology*, vol. 31, no. 6, pp. 506–516, 08 2016. [Online]. Available: https://doi.org/10.1093/arclin/acw049

[43] “Assistive Innovations - The improved iEAT Feeding Robot.” [Online]. Available: https://www.assistive-innovations.com/news/217-the-improved-ieat-feeding-robot

[44] “Bestic - en.” [Online]. Available: https://www.camanio.com/en/products/bestic/

[45] “Obi – The first dining robot of its kind.” [Online]. Available: https://meetobi.com/

[46] D. Gallenberger, T. Bhattacharjee, Y. Kim, and S. S. Srinivasa, “Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding,” in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2019, pp. 267–276. [Online]. Available: https://ieeexplore.ieee.org/document/8673309

[47] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa, “Towards Robotic Feeding: Role of Haptics in Fork-Based Food Manipulation,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1485–1492, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8624330

[48] “Autonomous robot feeding with assistive dexterous arm (ada).” [Online]. Available: https://www.youtube.com/watch?v=t2eO4CD-0WY

[49] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira, “Discriminative optimization: Theory and applications to computer vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 4, pp. 829–843, April 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8338139

[50] “Intel ® RealSense™ 3D Camera ZR300,” 2017. [Online]. Available: https://click.intel.com/media/ZR300-Product-Datasheet-Public-002.pdf

[51] “UP Specifications – UP Bridge the Gap.” [Online]. Available: https://up-board.org/up/specifications/

[52] A. Mordvintsev and A. K, “Camera Calibration — OpenCV documentation,” p. 1, 2016. [Online]. Available: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

A

Appendix A: Depth Cameras

To model how a camera works, i.e. how a view is projected into an image as shown in Fig. A.1, the pin-hole model Eq.(A.1) is used, where X,Y,Z correspond to the position 3D position while x and y correspond to the pixel coordinates, f_X f_y denote the focal length on the in $\mathrm{pixel/meter}$, the c_x and c_y correspond to the center position of the camera in pixel, and λ is a scaling factor. This model is used to project 3D points into a 2D image as shown in

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}^{\mathbf{K}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{A.1}$$

Figure A.1: Pinhole Camera Model. Image taken from[1].

More important than Eq.(A.1) are the opposite equations which allow the conversion from a depth image into 3D data. A depth image encodes in each pixel a depth value corresponding to how far in the image that pixel was captured. They are represented in Eq.(A.2) where $Z(u, v)$ corresponds to the depth readings at pixel coordinates u and v ,

$$X = \frac{Z(u, v)(u - c_x)}{f_x} \tag{A.2a}$$

$$Y = \frac{Z(u, v)(v - c_y)}{f_y} \tag{A.2b}$$

A distortion model can also be added on top of the pin-hole model, to take into account cameras where the lens may not be correctly aligned with the image plane, and even model radial distortion from lenses that create the fish-eye effect. There are currently four mains methods to obtain a depth image as shown in Fig. A.2.

[1] https://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/
[2] https://medium.com/@DAQRI/depth-cameras-for-mobile-ar-from-iphones-to-wearables-and-beyond-ea29758ec280

Figure A.2: Types of Depth Cameras. Taken from[2].

Passive & Active Stereo Cameras

Both passive and active stereo 3D cameras work in the same way, they consist of two cameras that form a stereo pair. Each of the cameras looks for features in its images which are then used to calculate a disparity image from which the depth of the sensor can be extracted. The only difference between the active and passive implementation is that the active stereo camera also includes a laser projector, that projects a texture image onto the scene as a way to create more features that each sensor can pick up. A very good property of this types of 3D camera is that adding more of them in the same area does not cause interference between them, since the passive stereo does not change anything in the scene and active stereo would project more textures onto the scene, which would help to capture even more features.

Structured Light Cameras

Structured Light cameras, work also through stereo, but they use just a camera and a projector. Some known patterns are projected onto the scene which the camera locates. Matchings are then found between the camera located pattern and the known projected ones which are used to calculate deoth. The downside of structured light devices in multi camera networks is that they can easily cause interference on each other, consequently degrading the quality of the depth image obtained.

Time of Flight Cameras

This type of sensors do not use stereo. They work by emitting light and capture how long it took to reach the sensor, measuring its ”time of flight”, and calculating distances from it. This types of sensors are not very prone to interference for uses with multiple Time of Flight cameras because, a specific wavelength can be used for each sensor, and also due to each sensor only needing a very short amount of time to take a measurement. One downside is that if some objects do not reflect light directly, the wrong distances are calculated.

B

System Description and Functionalities

Contents

B.1 Hardware 62
B.2 Software 62
B.3 Functionalities 63
B.4 Limitations 65
B.5 Scalability and Flexibility 66

The system developed in this thesis, was made to be easily scaled and altered, both hardware and software wise. A detailed explanation of the full system is given in this chapter.

B.1 Hardware

Regarding the system hardware this consists on a main server running Ubuntu 16.04 with an Intel® Core™ i7-4790 CPU @ 3.60GHz $\times$ 8 processor 8GB DDR3 RAM. The system is also composed by consumer grade 3D cameras coupled to a small factor computer for local processing, the number of this camera+computer stations varied between 3 to 5 on the performed tests. This architecture allows for a more distributed and easily scalable system. The used cameras are Intel RealSense ZR300 [50] 3D cameras shown in Fig. B.1. This sensor has a depth range between 0.5 and 2.8 meters, and uses IR laser projector in conjunction with stereo IR cameras to infer depth. Besides the color and depth cameras, the sensor also has an IMU (Inertial Measurement Unit) and a gray fisheye camera which were not used. Although it is possible to change the camera configuration parameters (Image Gain, Exposure, Saturation, Contrast etc) the factory default ones are used. Among these parameters were the frame rates used for the RGB and Depth video stream which were both set to 30 fps, and their resolutions which are 640x480px and 480x360px respectively. The small computer coupled to each camera is an Upboard [51], which has a Intel® ATOM™ x5-Z8350 Processors 64 bits @ 1.44GHz and 4GB of RAM. All of these computers are connected to a switch, named TL-SG108E, with up to gigabit speeds, thus creating a local network where all the computers can communicate through Ethernet. The network diagram can be seen in Fig. B.2.

(a) Intel RealSense ZR300.

(b) Intel RealSense ZR300 Sensors.

Figure B.1: Utilized 3D Camera.

B.2 Software

For the whole system to run and be easily updatable, a few preparations had to be done before running the calibration software developed. The code and detailed explanations for the software and extra configurations can be found in the project repositories [1],[2].

[1] https://github.com/DonHaul/ROS-IntelZR-300-CameraNetwork/
[2] https://github.com/DonHaul/MscThesis

Figure B.2: System Diagram.

B.2.1 Configurations

This system is based on Robot Operating System (ROS), a middleware that simplifies communications between machines. ROS was installed in each computer in the network and the server was set as the ROS master, where the main framework will run. The necessary library to access the data stream coming from each RGB-D camera was also added to each Upboard [3]. A clone of the repository[1], containing system configurations and useful scripts to start, stop, update and setup the system, was added to every machine, so that the whole system is easy to control and to update.

B.2.2 System Start and Stop

The whole system is started, by initializing a ROS client on the server, after that, it is communicated via SSH (Secure Shell) with each UpBoard, that they can start up the 3D sensors and transmit its data to the server where the main computations will be done. A similar procedure happens to stop the whole system, the ROS client is closed and a it is again communicated via SSH to each board to stop transmitting data.

B.3 Functionalities

B.3.1 System Calibration

The methods to obtain the calibration object model or the camera poses are described in sections 3.4 and 4.1 respectively. This methods are both ran through the same code, as the rotations and translations are obtained in similar manner. A simple input file is given to specify some configurations, such as the

[3] http://wiki.ros.org/realsense_camera

dimensions of the markers or the number of cameras for example and whether we wish to obtain the calibration object or the camera poses.

A better description of all configuration possibilities is available in the calibration repository [4].

Before starting the calibration process, the intrinsic parameters of the cameras must be known. Most vendors, such as Intel, have this parameters openly available to the public, saving us from having to calibrate the individual cameras. In the case that such parameters are not specified, an intrinsic and distortion parameter calibration must be performed beforehand using for instance Chessboard Calibration [52].

B.3.1.A Obtaining Calibration Object

To calibrate the object through the method in Section 3.4, the calibration object must be moved in front of a camera. To improve the marker detection, it is recommended for the calibration object to be moved as slowly and to be as close to the camera as possible as shown in Fig. B.3. On top of that the scene must be well lit, as in darker scenes, the quality of the detections will be worse. The process has two steps, first it obtains the rotations and after that it obtains the translations.

After they are obtained, we retrieve the transformations from each marker and its corner points relative to a world reference frame. This model can then be used to obtain the full calibration object pose in each cameras referential in Section B.3.1.B

Figure B.3: Calibration Object Calibration Setup.

B.3.1.B Obtaining Camera Poses

The whole camera network calibration has a very similar process to what is described in Section B.3.1.A. The calibration object must be waved in front of the cameras. By moving and rotating the calibration

[4] https://github.com/DonHaul/MultiCamCalAruco

object around the whole scene and with different orientations, we mitigate biases that may occur if it remained in the same spot. Again, it is recommended for it to be moved slowly to counter possible desynchronization issues between cameras and also so that motion blur is not introduced into the images difficulting the markers detection.

In the end, camera transformations in the world referential are retrieved. These transformations can then be fine tuned using the algorithm in 4.2.

B.3.2 Real-Time 3D Reconstruction

After obtaining the camera transformations, it is now possible to use them, to transform the 3D data from each camera to create a more complete 3D reconstruction of a scene. Using ROS and the tf package[5] we transmit those transformations and apply them to the existing point clouds, obtaining a real-time reconstruction of the scene.

B.3.3 Point Cloud Cropping

Another feature that was added was the possibility to define a 3D Region of interest, i.e. a box, around the area that we wish to oversee, thus removing all point cloud data out of it. This cleans up the data and also prevents the transmission of unnecessary information through the network. This works by broadcasting a bounding box in a certain reference frame from the server into every UpBoard. Then each UpBoard transforms it into the associated camera's local reference frame, allowing them to trim the unwanted points data points locally.

B.4 Limitations

In the current system, to obtain the initial camera poses, most of the processing is done on the main server. This means that when we want to fetch the camera transformations, each individual camera image stream is sent to the server where the processing occurs. This may be problematic for systems with a larger number of cameras. Instead of sending image streams, an alternative is to detect the calibration object in each individual UpBoard, and then send only a stream of the calibration object's pose, which would greatly reduce the network bandwidth usage. Currently, the formula to calculate the bit rate of all the infromation that flows to the main server is presented in

$$bitrate_{img} = N \cdot w \cdot h \cdot 3 \cdot 8 \cdot FPS, \tag{B.1}$$

[5] http://wiki.ros.org/tf

where N corresponds to the number of cameras, the w and the h correspond to the width and height of the current image size of every camera. 3 corresponds to the number of color channels (RGB), 8 corresponds to the size of each channel color encoding, in this case each channel will have a value between 0 and 255 which means an 8 bit integer is needed to encode it, and finally FPS corresponds to the Frames Per Second of every camera.

If we were to use the pose stream instead the bit rate would be

$$bitrate_{pose} = N(64 \cdot 3 + 64 \cdot 4) \cdot FPS, \tag{B.2}$$

where the ROS pose message[6] is assumed to be used. The $(64 \cdot 3)$ corresponds to the bits used to encode the translation , since a 64bit float is used for each dimension (x,y,z). $(64 \cdot 4)$ corresponds to the bits to encode the rotation, here quaternions are used as specified by the ROS pose message. A quaternion has 4 dimensions (x,y,z,w) and each of the uses a 64bit float to encode its value.

From the formulas (B.1) and (B.2) we understand that by streaming poses, it is possible to have a system with more cameras, higher image quality, and higher frame rates, by a fraction of the current systems bandwidth. For the used cameras and frame rates, the current system with the current cameras uses around 17600 more bandwidth than if just poses were sent.

Another issue with the current system is that the cameras get easily desynchronized which results in bad pose estimations unless the calibration object is either static or slowly moving.

Finally a design simplification was the use of rectified images, i.e. images that have no distortion. This means that images from cameras with some distortion (radial or tangential) must be undistorted beforehand.

B.5 Scalability and Flexibility

This current system can be easily scaled up as it is fairly simple to configure a new 3D camera + UpBoard pair, and add them to the network. Furthermore it is possible to use the system with other commercially available 3D cameras and also combinations of different cameras, as long as there is a ROS package made by the manufacturer to integrate them with the built framework. Finally, it is also possible to use just the calibration framework developed without using ROS nor hardware. RGB image files can be fed into the system, instead of the real-time images from each camera.

[6] http://docs.ros.org/melodic/api/geometry_msgs/html/msg/Pose.html