Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Filipe Oliveira Costa

Calibração de Sensores do ATLASCAR2 por Otimização Global

ATLASCAR2 Sensors Calibration by Global Optimization

Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Filipe Oliveira Costa

Calibração de Sensores do ATLASCAR2 por Otimização Global

ATLASCAR2 Sensors Calibration by Global Optimization

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Miguel Armando Riem de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Vítor Manuel Ferreira dos Santos, Professor Associado com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Agradecimentos / Acknowledgements

Em primeiro lugar gostaria de agradecer ao Professor Miguel Oliveira pela orientação neste trabalho, onde foi notável a dedicação e motivação que o professor demonstrou, onde nunca deixou que eu perdesse o rumo ao objetivo final.

Quero também agradecer ao Professor Vitor Santos pois procurou sempre acompanhar o trabalho, demonstrando interesse e disponibilizando-se sempre a ajudar.

Quero também deixar um agradecimento ao Professor Paulo Dias por emprestar o equipamento ASUS ZenFone AR para que fosse possível a realização dos ensaios.

Quero deixar um grande obrigado à minha família pois sem eles isto nunca teria sido possível, pois apoiaram-me nos momentos bons e menos bons de todo este percurso, nunca me negando a ajuda.

Por fim quero agradecer a todos os meus colegas que viveram comigo este trajeto, onde mostraram sempre companheirismo e entreajuda, onde nunca deixaram um soldado para trás.

Palavras-chave

calibração de cameras; fusão de sensores; calibração extrínseca; visão computacional; marcadores Aruco; otimização.

Resumo

Em veículos autónomos é frequente a necessidade de instalar um grande número de sensores a bordo. Assim, a calibração extrínseca destes sistemas multi-sensoriais é um problema de grande relevância para o desenvolvimento de algoritmos de condução autónoma ou de apoio à condução. Este trabalho propõe um mecanismo capaz de fazer uma calibração automática em simultâneo de várias câmaras. No processo são usados marcadores aruco, o que permite estabelecer um grafo de onde se extraem as transformações geométricas entre as várias câmaras e um referencial global. Inicialmente, os marcadores são detetados nas imagens usando uma ferramenta do OpenCV. Posteriormente é construido o grafo em que os nós são câmaras ou marcadores, e as ligações entre nós são transformações geométricas em pares câmara aruco. Em seguida é calculada uma estimativa inicial dos parâmetros extrínsecos de todas as câmaras, baseada nas deteções dos marcadores e nos caminhos obtidos do grafo. No fim, é feita uma otimização dos parâmetros, onde é minimizado o erro de reprojeção. Para demonstrar o processo foram criados vários "datasets", de modo a validar os resultados obtidos.

Keywords camera calibration; sensor fusion; extrinsic calibration; computational vision; Aruco markers; optimization.

Abstract In autonomous vehicles, it is often necessary to install a large number of sensors on board. Thus, the extrinsic calibration of these multi-sensory systems is a problem of high relevance for the development algorithms of autonomous driving or of assistance to the driving. This work proposes a tool to automatically calibrate simultaneously multiple cameras. In the process, aruco markers are used, which allows establishing a graph from which the geometric transformations between the various cameras and a global reference are extracted. Initially, markers are detected in the images using an OpenCV tool. Subsequently, the graph is established where the nodes are cameras or markers and the edges are the transformations between them. Then an initial estimate of the extrinsic parameters of all cameras is calculated based on the detections of the markers and the paths obtained from the graph. In the end, an optimization of the parameters is done, where the reprojection error is minimized. In order to demonstrate the process, several datasets were created in order to validate the obtained results.

Contents

List of Tables

List of Figures

Acronyms

AD Autonomous Driving. 1

ADAS Advanced Driver Assistance Systems. 1

BA Bundle Adjustment. 18

CenSurE Center Surround Extremas. 17

HSV Hue, Saturation, and Value. 14

IAV Internal Angle Variance. 10

ICP Iterative Closest Point. 10, 12, 16

IEPF Iterative-End-Point-Fit. 13

LAR Laboratory for Automation and Robotics. 5, 14

LIDAR Light Detection And Ranging. 2

LRF Laser Rangefinder. v, 4–14, 16, 58

MI Mutual Information. 13

OpenCV Open Source Computer Vision Library. 12, 14, 16, 17, 21, 22, 25, 27, 28, 57

PCL Point Cloud Library. 9, 10

RANSAC Random Sample and Consensus. 7, 9, 10, 17

ROS Robot Operating System. 21

SBA Sparse Bundle Adjustment. 18

SciPy Scientific computing in Python. 21, 22, 36

SFM Structure-from-Motion. 18

SNN Spatial Nearest Neighbour. 9

trf Trust Region Reflective. 36

Chapter 1

Introduction

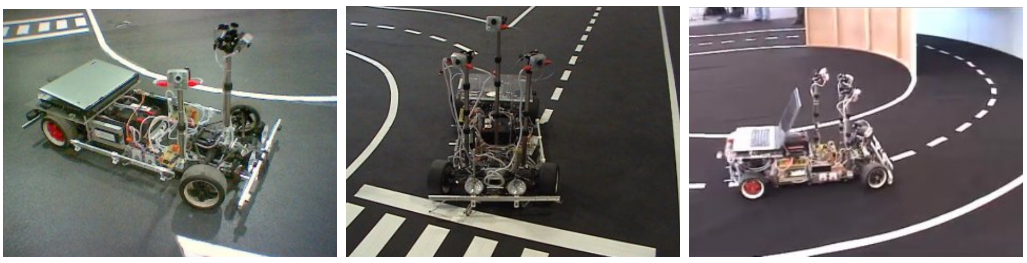
Nowadays, most cars rely on Advanced Driver Assistance Systems (ADAS) to avoid vehicle crashes, to park automatically, to recognize traffic signs and sensors to give the driver perception of the surrounding area [33]. All this technology proves that Autonomous Driving (AD) is no longer science fiction. Some companies, like Google, Mercedes-Benz, Tesla and others, already have autonomous cars working, however they are still at level 2 of the AD level [49], which means that the car can steer and accelerate or decelerate using information about the driving environment, but the driver must be ready to operate the pedals and steering wheel if there is a fault in the system.

There are many techniques used in AD, and each one has a different way of functioning. In any case, these methods use a great number of sensors on the car with the purpose of improving the quality of the data. Using many sensors involves a calibration process which merges all the information taken from the sensors and thus has a real notion of the space to be interpreted.

Multi-sensor platforms are not just applications for AD. These can also be used for the 3D reconstruction of buildings, where this reconstruction can be used for the rehabilitation of old buildings that have degraded over the years. Platforms can also make 3D models in the field of biology which makes it easier to study them. All this is possible with a good calibration of the sensors in order to allow the data fusion from the sensors.

In the literature, there is a multiplicity of solutions to carry out the calibration of the sensors. However, there is not yet an optimal solution which can calibrate more than two sensors automatically, without user intervention. The implemented methods usually require user intervention to select matching points such as the calibration target and some even need good lighting conditions to work. All these factors induce errors in the calibration process, and prevent the calibration process from being automatic. Calibrating a set of sensors at a random place and without any human intervention would be the ideal solution of a calibration process.

This document presents a new approach to calibrate N cameras using fiducial markers as a reference pattern. The method begins by obtaining an initial estimation of the cameras' position and the position of the markers, and then using the bundle adjustment method, it is possible to minimize the reprojection errors and output the optimal solution for the extrinsic parameters of the cameras.



(a) (b) (c)

Figure 1.1: Some of the ATLAS project small scale platforms [2].

1.1 ATLAS project

ATLAS is a project created by a Group from Automation and Robotics in the Department of Mechanical Engineering at the University of Aveiro [47]. The goal of this project is to develop systems to promote the autonomous control of cars and other platforms. Firstly, the ATLAS project started to build up systems to allow autonomous navigation using small scale platforms in controlled environments. The ATLAS platforms have participated in many competitions winning some awards for the best performance. Some platforms are represented in Fig. 1.1.

In 2010, the ATLAS project migrated from small scale platforms to a common car (Fig. 1.2), in order to start a new challenge in autonomous driving. This new challenge is named ATLASCAR and presents greater complications, in relation to the perception of the surrounding area, since the roads of circulation do not always have the same characteristics. The lighting conditions represent another major complication, as they are completely random due to the climatic conditions. Initially, some hardware was applied to the car: an active vision unit, a thermal camera, a 3D Light Detection And Ranging (LIDAR) and a stereo camera, to allow a robust perception from the external environment and agents. Some research was done with this car, where it was possible that the car followed a person. This was an important milestone for the autonomous driving of ATLASCAR.

In 2017, the ATLASCAR project migrated to a new and more modern car with electric motorization (Fig. 1.3). This ATLASCAR2 already has some equipment installed on board that migrated from the previous ATLASCAR, as well as some new equipment.

1.2 Project context

In autonomous driving there is the need to make quick decisions because the car is moving at high speeds and a wrong or even slow decision can be fatal to the performance of the car. For this, several sensors are used which allow the car to know the surrounding space and make the detection of people or objects in motion, and thus decide which direction should to take, or even accelerate or stop.

As mentioned earlier, ATLASCAR2 is the design of a self-contained car and therefore contains some installed sensors. Currently, the ATLASCAR2 is equipped with two Sick LMS151 which is a robust 2D LIDAR, one Sick LD-MRS400001 which is a 3D LIDAR and one Point Grey ZBR2-PGEHD-20S4C camera.

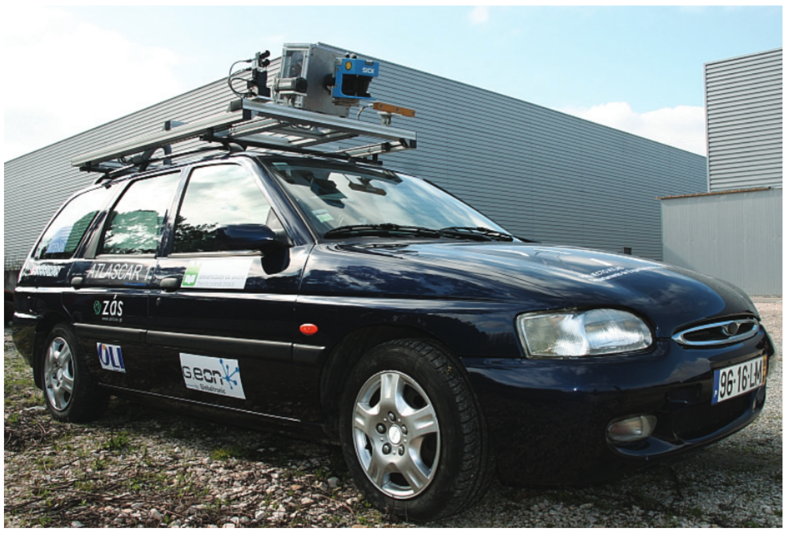


Figure 1.2: The car used in ATLASCAR [47].

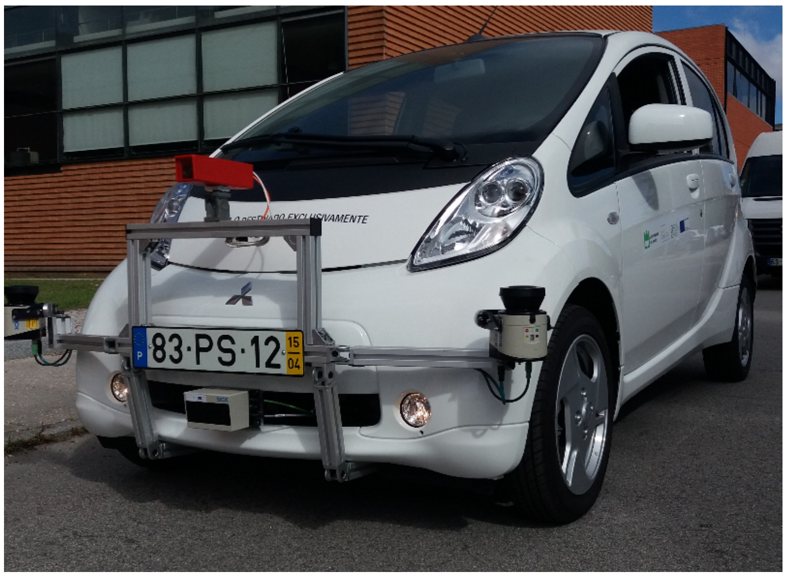


Figure 1.3: The car used in ATLASCAR2.

For all these sensors to be really useful, it is important to perform their extrinsic calibration, to have a better perception of the surrounding area. Due to the variety of the existing sensors on the car, the calibration process becomes very complex.

The extrinsic calibration of all the sensors that are on board ATLASCAR will allow us to know their position in relation to a common reference, which will allow the fusion of all the data obtained through the sensors. The merging of the data obtained from the sensors is important because it allows the analysis of the data as one and cross-references the information of all sensors, thus allowing us to acquire a good perception of the surrounding scene.

In previous years, Pereira [39] and Silva [51] developed an approach to calibrate some sensors using a ball as a calibration target, which is a good calibration target because is easily detected both by cameras as well as by LRFs. However, the approach used is not totally automatic because it calibrates the sensors in pairs, where all sensors are calibrated relative to one initially chosen to be the reference sensor. This method presents good results for the calibration of LRFs. However, it has some problems with the calibration of stereo cameras.

1.3 Objective

In order to carry out more advanced studies of driving perception and assistance, it is necessary to calibrate the various sensors on board of the vehicle. The calibration process is a sporadic procedure that aims to estimate the geometric transformations between each pair of sensors on board the car. These transformations are later used to map data from one sensor to another. The goal of this work is to investigate the possibility of using the global optimization processes to perform a calibration of the sensors on board of the ATLASCAR2.

The global optimization consists of a process that seeks to minimize the error of reprojection of the data obtained from the sensors. This optimization seeks to vary the results obtained from an initial calibration in order to lower reprojection error, if possible up to zero. In the data to be optimized, the extrinsic parameters of the sensors as well as the positions of the points detected by the sensors are present. Intrinsic parameters can also be inserted in the optimization if the sensors are cameras. Due to the variety of parameters that the optimizer varies to find a more precise solution, both extrinsic and intrinsic parameters, and the position of the points obtained by the sensors, this process is called global optimization.

1.4 Document structure

This document is divided into six chapters where a brief introduction was initially made and the objectives for the development of this work were also presented. Then, in chapter 2, some work related to the calibration of sensors mentioned in the literature is presented, where LRF to LRF calibration, camera to LRF calibration and camera to camera calibration methods are presented. Some optimization methods are also mentioned. After, in chapter 3 the software and hardware used in the accomplishment of this work will be presented. In chapter 4, the calibration method developed in this work will be presented, where the calibration target used will be mentioned, followed by the calibration process that is carried out. In chapter 5 the datasets that were elaborated to test the implemented calibration method as well as the results that were obtained are exposed. Finally, in chapter 6, all the conclusions that were obtained during the accomplishment of this work are detailed and to conclude some future work is proposed that will be able to give continuity to this new method of calibration.

Chapter 2

Related work

Multisensor platforms provide a large amount of data, but the utility of this information depends on the possibility to merge all this data in a common reference frame. This process of merging requires extrinsic calibration of the sensors. Over years, many solutions have been presented. However, today there is not a great solution to this problem which calibrates all type of sensors at the same time, without manual human intervention. In this chapter, a brief explanation of the extrinsic and intrinsic calibration is presented, and also, some calibration methods proposed from the literature, as well as the calibration method implemented in Laboratory for Automation and Robotics (LAR) to calibrate the ATLASCAR sensors.

Before presenting the several methods of calibration used in the state of the art, it is important to understand the concept of extrinsic calibration. Extrinsic calibration is the process of finding the relative rotation and translation between different coordinate systems. Each coordinate system refers to a sensor which is to be calibrated. The translation and rotation combined are often referred to as transformation. Whenever a new sensor is added to the system or the placement of the sensors changes, these transformations must be recomputed by a process designated extrinsic calibration.

As previously mentioned, the transformation is a combination of a rotation component and a translation component, which results in the matrix represented on the Eq. 2.1. The transformation matrix T is composed by a 3×3 matrix (r - rotation component), and a 3×1 matrix (t - translation component).

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

The transformation between two coordinate systems is represented by ${}^{R}T_{N}$. If the coordinate system R is a calibration reference frame and the coordinate system N is a Laser Rangefinder (LRF) coordinate system, the extrinsic matrix of the LRF is the transformation matrix ${}^{R}T_{N}$, as illustrated in Fig. 2.1.

To move a point defined in a coordinate system N to a coordinate system R, the transformation that will be applied to the point is the transformation ${}^{R}T_{N}$.

Figure 2.2 shows a process of extrinsic calibration. The process of extrinsic calibration consists of searching for the values of the transformation between two sensors that

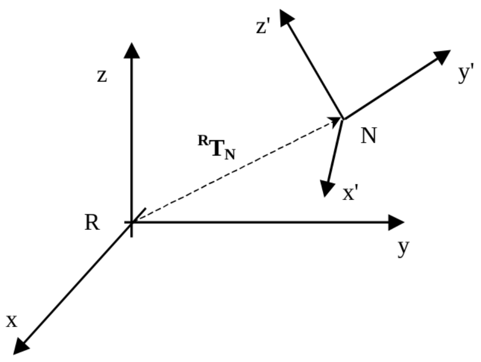
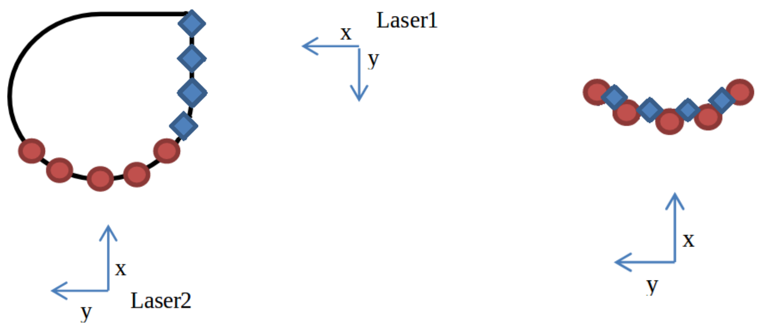


Figure 2.1: ${}^{R}T_{N}$: The relation between two coordinate systems [48].



(a) (b) Overlap of the LRF 1 and 2

Figure 2.2: Data acquisition from two 2D LRF (Uncalibrated) [9].

give the best quality overlap between the data viewed by the sensors, in order to allow for a good reconstruction of the scenario.

In Fig. 2.2 (a) two LRFs are represented in different positions. The blue diamonds and red spheres represent the data acquired by LRF 1 and LRF 2 respectively, which are represented on the respective LRF reference frame. Without performing any calibration process, the result of overlapping the data of the two LRFs has the result shown in the Fig. 2.2 (b). This result does not represent any useful information since it does not allow for the reconstruction of the object as it is represented in the Fig. 2.2 (a). For this, it is necessary to calibrate the sensors in relation to a common reference and with the calibrated sensors it is possible to determine the side of the object that is being observed by each LRF. Thereby it is possible to rebuild the object.

2.1 LRF to LRF calibration

LRF sensors are a technology widely used by robots and intelligent vehicles in order to generate a perception of the surrounding area. However, when many LRFs are used to have the perception of the same object, it is necessary to calibrate them. This calibration

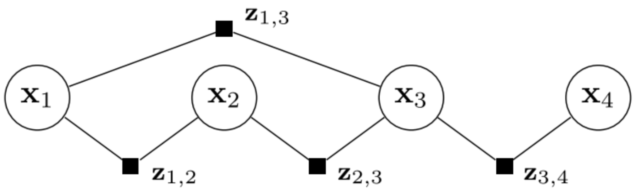


Figure 2.3: Transformation graph where x_i represents the LRFs and z_i represents the transformations between the pair of LRFs [45].

consists of estimating the extrinsic parameters of the LRFs, i.e. to estimate the position of the LRF sensors in relation to a common reference frame.

In 2015, Rowekamper et. al. [45] proposed a new approach to calibrate a network of multiple planar LRF with horizontal scanning. The implemented method can estimate the extrinsic parameters of all LRF sensors at the same time and does not need a specific calibration pattern like a chessboard. The method only needs an obstacle that moves in front of all sensors, to get an initial estimation of the extrinsic parameters of the LRFs.

In order to acquire an initial estimation of the extrinsic parameters, the authors begin by detecting the obstacle in motion by all LRF sensors. For that, the authors use a background subtraction technique [10], that consists of subtracting the actual data from the previous data acquired by the LRF sensor and to detect the differences. The detected differences correspond to the obstacle that is in motion. Because the LRFs do not see the obstacle at the same time, the authors use a Random Sample and Consensus (RANSAC) algorithm to estimate the transformations between each pair of LRFs that see the obstacle at the same time.

RANSAC was proposed by Fischler and Bolles [18] in 1981. This algorithm is an iterative method to estimate the parameters of a mathematical model using the minimum amount of information. This algorithm begins by using the smallest set of data and then complements it with more information until it obtains the best possible result for the parameters.

As the extrinsic calibration consists of getting the positions of all sensors with respect to a common reference frame, and the position estimation of the sensors acquired previously only refers the position between a pair of LRFs, the authors built a node graph (Fig. 2.3) where each node represents one sensor and the connections between nodes refer to the estimated pairwise transformations. In this way, authors compute the transformation of all sensors in relation to a common reference frame, giving more weight to the connections with more transformations.

Finally, to make the calibration of all LRF sensors, the authors add the point correspondences observed by the LRFs to the node graph, as is illustrated in Fig. 2.4. Then, they do an optimization based on the bundle adjustment method [50], seeking to minimize the reprojection error of the transformations of the LRF pairs and all point correspondences observed by LRFs.

The authors did two tests to evaluate the method. In the first test (Fig. 2.5), they varied the height position of the LRFs, and in the second test (Fig. 2.6) they put all LRFs in the same height position. They concluded that the implemented method does not have good results for the first test, where they obtained a consistency score of

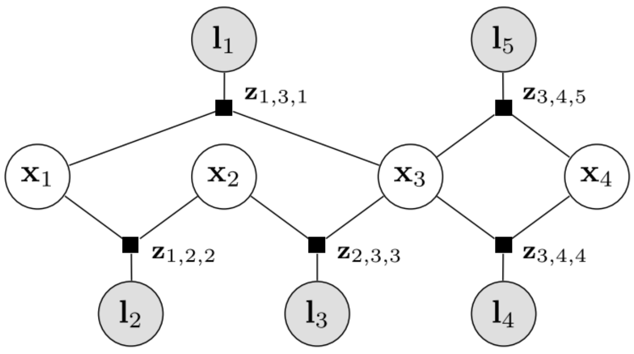


Figure 2.4: Transformation graph where x_i represents the LRFs and z_i represents the transformations between the pair of LRFs and l_i represents the point correspondences observed by the LRFs [45].

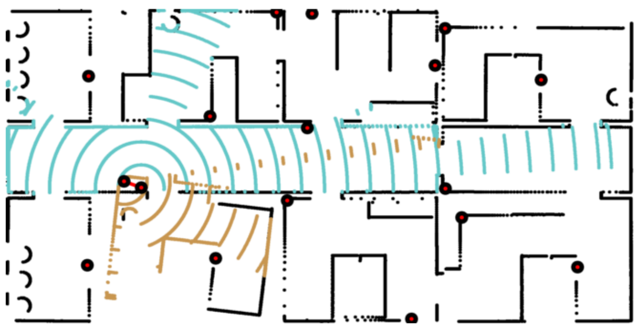


Figure 2.5: Results of first test of the calibration method implemented in [45].

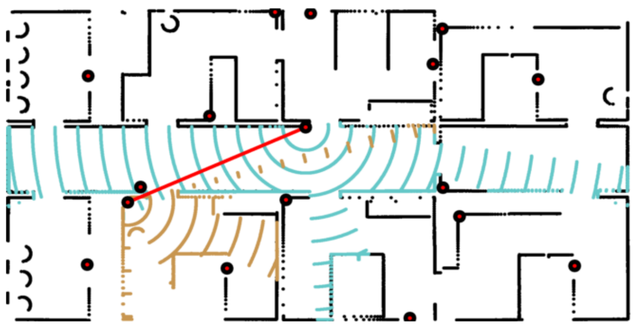


Figure 2.6: Results of second test of the calibration method implemented in [45].

24%. However, in the second test they obtained a consistency score of 92%, which they consider a good result and which they argue, validates the proposed method.

Fernandez-Moral et. al. presented a calibration of a 2D LRF method from perpendicular plane observation [17]. This method consists of establishing geometric constraints

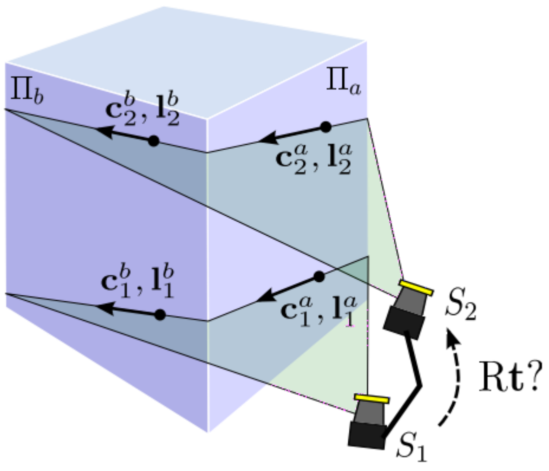

Π_b
Π_a
$\mathbf{c}_2^b, \mathbf{l}_2^b$
$\mathbf{c}_2^a, \mathbf{l}_2^a$
$\mathbf{c}_1^b, \mathbf{l}_1^b$
$\mathbf{c}_1^a, \mathbf{l}_1^a$
S_2
Rt?
S_1

Figure 2.7: Observation of a corner structure (two perpendicular planes) by two LRFs [17].

between the observations of pairs of perpendicular planes, which is shown in Fig. 2.7.

The proposed method needs a first approximation for the position of the sensors, and the measurement planes of at least two sensors must intersect. This method calibrates the LRF sensors in pairs, and also needs to have at least two observations corners from different orientations, so that it may calibrate a pair of sensors. This approach begins with the detection of the corners by the LRFs, where the corners detected are selected through co-planarity and orthogonality constraints. After corner selection, a RANSAC method to estimate the transformations between a pair of sensors is used. The method will add and remove detected corners until it achieves a good solution.

Pereira presented one method to do the extrinsic calibration of the LRF sensors of the ATLASCAR [39], [37], [38]. This method uses a ball as a calibration pattern, which has the advantage of being detectable both by LRFs as well as by cameras. This method of calibration requires the diameter of the ball to be known.

The method of calibration consists of detecting the center of the ball by all sensors in many positions, and generating a point cloud with all of the detected ball centers for each sensor, as is illustrated in Fig. 2.8. Finally, one of the LRF coordinate systems is chosen to be the reference for calibration, in order to calibrate all sensors. Then the LRFs are calibrated one at a time in relation to the calibration frame using an algorithm of the Point Cloud Library (PCL) [46].

The detection of the ball and its center is one of the biggest problems in this calibration method. This problem has different difficulties depending on the type of data. In 2D data, the author begins with a segmentation of the LRF scan, which consists of clustering sets of points with a high probability of belonging to the ball. For this, the author uses the Spatial Nearest Neighbour (SNN), based on [14], which is a recursive algorithm that computes the Euclidean distance (Eq. 2.2) between one point and all other points, and when the computed distance is smaller than a threshold, this point is clustered to the interest points. After that, the author uses a method to detect the

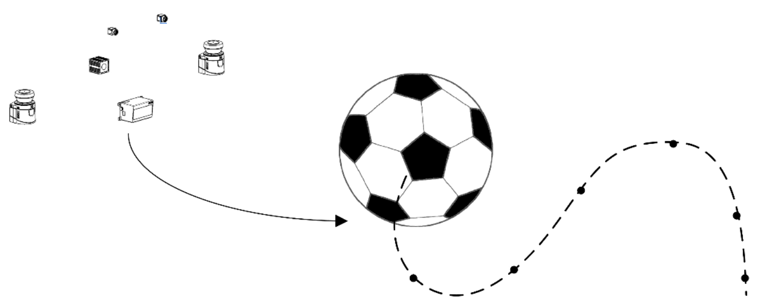


Figure 2.8: Overview of the calibration approach by Pereira [39].

circle, based on [53], which uses a technique named Internal Angle Variance (IAV). IAV uses the trigonometric properties of the arcs, where every point in the arc has congruent angles in respect to the extremes. Then the author determines the circle proprieties: the coordinates center and radius. Finally, it computes the center of the ball given its diameter, using trigonometric relations.

$$d_{E,N_{\text{pontos}}} = \sum_{n=1}^{N} \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \tag{2.2}$$

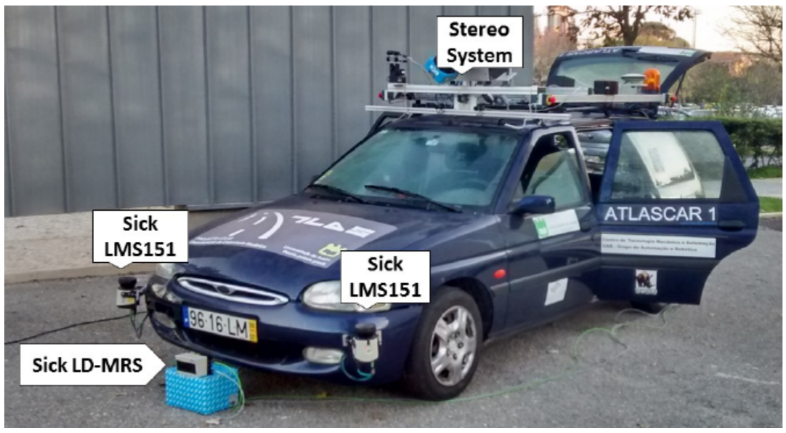
In 3D data, the author uses the PCL segmentation capabilities, namely the sample consensus module, that uses a RANSAC estimator method. This model accepts as input a point cloud and outputs the coordinates of the center of the ball and the radius.

To estimate the transformations, the author uses a variant of the Iterative Closest Point (ICP) algorithm, available on PCL, which is used to estimate the transformation between two point clouds. This variant has two error minimization metrics, and the author selected the point-to-point metric, which minimizes the sum of the Euclidean distance between corresponding points.

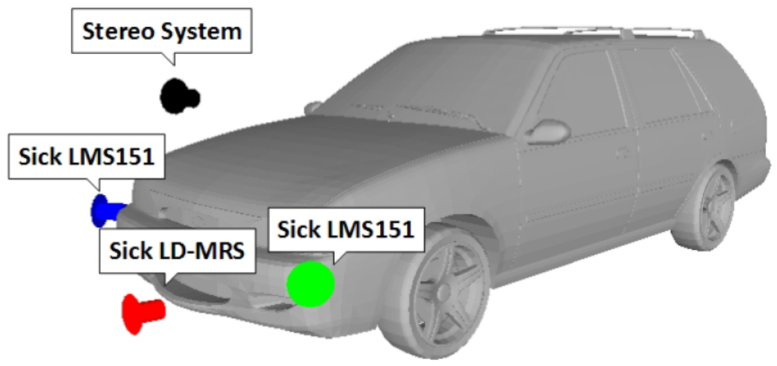
The authors tested the method on sensors of ATLASCAR1, and concluded that the calibration results were accurate, with the exception of the stereo system which yielded a small error. The results are represented in Fig. 2.9.

In 2016, Quenzel et. al. [41] used the calibration method implemented by Rowekamper et. al [45] previously mentioned, and made some modifications with the goal of increasing the robustness. The author did just two modifications. The first was to consider more objects in motion and not just one, to estimate the transformations between a pair of LRFs with the RANSAC method. The second difference was optimizing just the transformations between sensors and not the point correspondences observed by the LRFs. The authors claim that this method is more robust stating they had better results than Rowekamper et. al, and which are represented in Fig. 2.10.

Despite having good results, the aforementioned methods are not sufficient for our purposes because of the reasons presented below. In [45] and [41], the calibration requires that all LRFs are positioned at the same height, with co-planar scanning planes. In [17] and [39], the calibration is done in pairs of sensors. The fact that the calibration is done in pairs may overlook some relevant data. For example, if a calibration is performed between three sensors, 1,2 and 3, a pairwise calibration will first calibrate between sensor 1 and sensor 2 and then between 2 and 3. However, there may be useful information

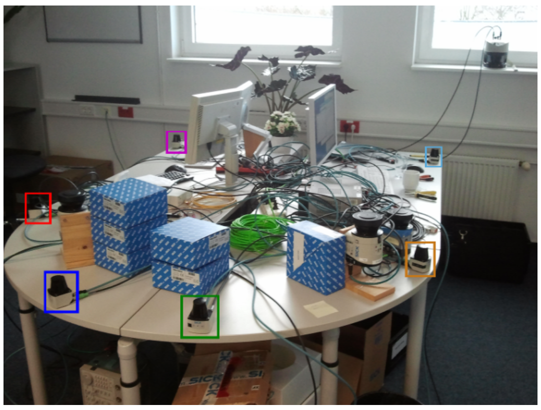


(a) Sensors setup on the ATLASCAR1.

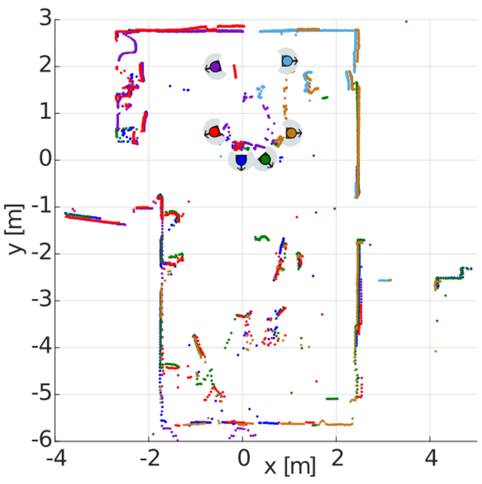


(b) Representation of the position sensors after the process of calibration.

Figure 2.9: Results of the calibration method developed in [39].



(a) Real position of the sensors LRF.



(b) Final result of the calibration process.

Figure 2.10: Results of the calibration method by [41].

between sensor 1 and sensor 3 which is discarded by pairwise calibration schemes. Note that only a global calibration methodology is able to address the problem described before, since it will take into account all the available data between any combination of sensor pairs. The method presented in [17] needs an initial estimation of the position of the sensors. This, however, is a common requirement in any optimization algorithm in order to avoid local minima. The method presented in [39] uses the ICP algorithm for estimating the pose of the sensors. This algorithm uses a pre-defined cost function which is not customizable.

2.2 Camera to LRF calibration

LRF sensors have good precision and are able to measure the surrounding area. Sometimes, it is difficult to know what the LRF sensors see. Therefore, adding a camera can solve this problem. However, to use a camera, besides the extrinsic calibration, it is necessary to make the intrinsic calibration and get the lens distortion. Nowadays, there are some shelf tools which get these parameters, like the Matlab Camera Calibration Toolbox [11] and also an algorithm implemented in Open Source Computer Vision Library (OpenCV) [15].

The two major distortions are radial distortion, which contain up to six coefficients usually denominated k_1, k_2, k_3, k_4, k_5 and k_6 and tangential distortion, which contains two coefficients denoted p_1 and p_2. To obtain the undistorted image, Eqs. (2.3) are applied to remove the radial distortion and the Eqs. 2.4 to remove the tangential distortion.

$$\begin{aligned} x_{\text{radial}} &= x \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{\text{radial}} &= y \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.3}$$

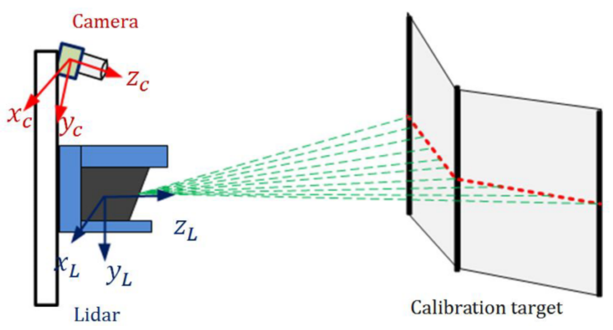
$$\begin{aligned} x_{\text{tangential}} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{\text{tangential}} &= y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{aligned} \tag{2.4}$$

The intrinsic calibration consists of getting the intrinsic matrix, named camera matrix. The matrix K is represented in Eq. 2.5, where the f_x, f_y represent the focal length and the c_x, c_y represent the optical centers.

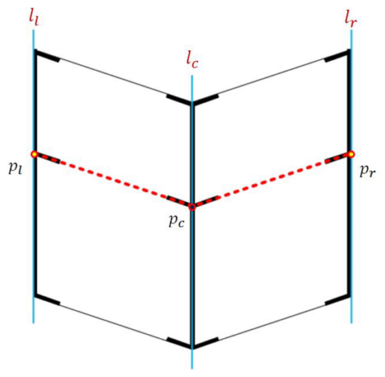
$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

In 2011, Kwak et. al. presented a new approach to carry out the calibration of 2D LRFs and cameras [28]. The method uses two planar boards arranged in a v-shape with an angle of 150° as a reference pattern, as is illustrated in Fig. 2.11(a). The convex side is used for calibration. In order to allow for better visualization by the camera, the author puts black tape on the lateral extremes and in the middle of the two planar boards.

The method begins by getting a set of images and LRF scans with the target in different poses and at different ranges. The LRF and camera have to see the reference pattern, ensuring that the LRF intersects the left and right sides of the target.



(a) Setup configuration.



(b) Calibration board description.

Figure 2.11: Camera and 2D LRF calibration using two planar boards arranged in a v-shape [28].

After data acquisition, the authors select three lines (l_l, l_c, l_r) and two points (p_l, p_r) manually from each image, as represented in Fig. 2.11(b). Then, knowing the dimensions of the target, the authors estimate the segments that link the points p_l and p_r with the center point p_c, using the Iterative-End-Point-Fit (IEPF) algorithm [16], that recursively splits a set of points until a distance related criterion is satisfied. Fitting lines to each segment using the total least squares method [21], the authors find the center point (p_c).

Finally, the authors estimate the extrinsic parameters by minimizing the distance between the line features and point features in 2D. Then, they project the LRF points into the camera image, and by an optimization problem they minimize the distance between points using the error function E represented in Eq. 2.7.

$$d_k^i = dist(p_k^i, l_k^i), k = left, center, right \tag{2.6}$$

$$E(R,t) = \sum_{i=1}^{N} (d_l^i)^2 + (d_c^i)^2 + (d_r^i)^2, \tag{2.7}$$

where the function $dist$ is the minimum distance from point p_k^i to line l_k^i and N the number of sets of images and LRF scans.

The authors perform experiments with 50, 100 and 150 image/scan pairs and get an error, respectively, of 6.0, 4.0 and 3.7 pixels. They compare the results with other more robust methods and claim that the alignment accuracy is better and also concluded too that the method needs fewer image/scan pairs to achieve the same calibration accuracy.

In 2012, Pandey et. al. presented a method to make the extrinsic calibration of a 3D LRF scanner and optical camera system [36]. The method does not need a reference pattern, and uses a Mutual Information (MI) framework which uses the surface reflectivity values reported by the range sensor and the grayscale intensity values reported by the camera to make the extrinsic calibration between these sensors. The authors say that the correct extrinsic calibration results from the maximization of the correlation between the LRF reflectivity and camera intensity. However, if the environment contains coloured surfaces that completely absorb infra-red light, these surfaces will show up as black patches in the LRF reflectivity and will be completely uncorrelated with the corresponding grayscale values obtained from the camera. Therefore, the mutual information based calibration technique might not work well in such situations.

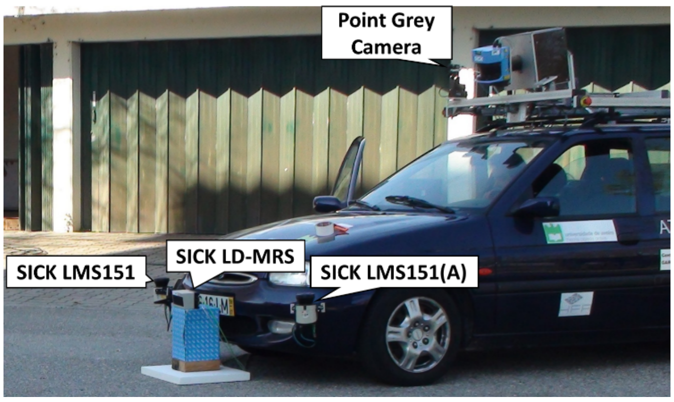
To complement the method of calibration, using a ball, developed in LAR, Silva implemented a camera to LRF calibration method, which uses a ball as a reference pattern [51], [38]. To estimate the transformation between sensors, the author tested two methods. The 3D Rigid Body Transformation method developed in [39] is used by the author but in this case the point cloud results from the conversion of the interest points on the image into 3D points in the camera coordinate system. The author developed an Extrinsic Camera Calibration method which instead of projecting the image points into a 3D coordinate system of the camera, it projects the 3D points detected by the LRF, into the 2D image projection. The problem is solved resorting to an iterative method based on Levenberg-Marquardt optimization algorithm [29], [32], finding a pose that minimizes reprojection error.

To detect the ball with a camera, the author uses an algorithm named *"Approximated Contour"* that uses the OpenCV libraries and follows three steps, 1) Finding contours in the image, where the contours are a set of points that represent a closed curve in an image; 2) Contour approximation by a polygonal curve; 3) Ball contour detection and circle properties, that select the contour that represents the ball. This algorithm only requires a Hue, Saturation, and Value (HSV) color range for thresholding. The HSV values change with the light conditions, thus the method requires these values to be set at the beginning of the calibration process. The results are illustrated in Fig. 2.12. The author concludes that the 3D Rigid Body Transformation method has a large error but the extrinsic camera calibration method has close results from real values.

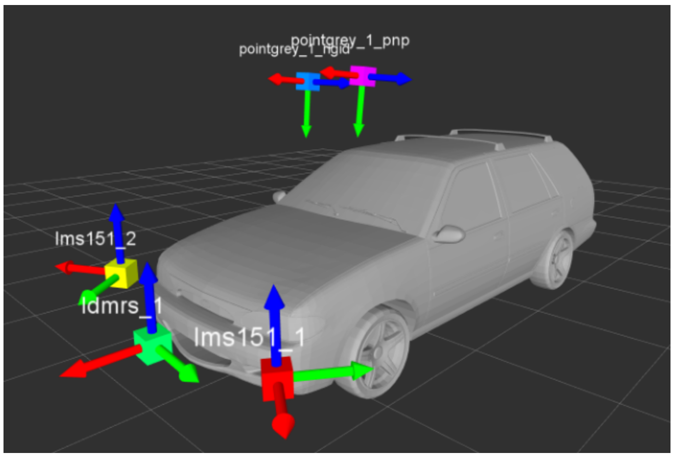
In 2017, Guindel et. al. presented a calibration method for a LRF sensor and a stereo system [22]. This method uses a planar board with four circular holes symmetrically disposed as a calibration pattern, as shown in Fig. 2.13. The approach of this method begins with the segmentation of the reference points in both point clouds, and then the process to estimate the extrinsic parameters. In order to compare the data acquired by the sensors, the authors need the data sensors to have a format like a point cloud. The LRF data is already a point cloud, however, the stereo data is represented by a grayscale intensity and depth estimation. To solve this, the authors use a stereo matching algorithm, which converts every pixel in a 3D point cloud. Having the point clouds it is necessary to make the segmentation of the clouds to recognize the calibration target, for this, the authors implemented a sample consensus-based segmentation method to determine the plane models, in each cloud, which the results of which are represented in Figs. 2.14(a) and 2.15(a). Then, the authors use the method implemented in [30], to find the points that correspond to discontinuities, on the LRF cloud, the result of which is represented in Fig. 2.14(b). For the camera, they applied a Sobel filter that results in Fig. 2.15(b).

The objective of this segmentation is getting the center points of the holes. With the objective of detecting the circles, the method filters the points resulting from the previous process concluding in Figs. 2.14(c) and 2.15(c). Having the circles, they find their centers (Figs. 2.14(d) and 2.15(d)), and finally the segmentation is ready to estimate the extrinsic parameters.

Finally, the authors find the initial transformation parameters by minimization of the distance between the reference points of both point clouds. Then, they use the ICP



(a) Sensors setup on the ATLASCAR1.



(b) Representation of sensors position after calibration process.

Figure 2.12: Results of calibration method developed in [51]. Point Grey camera using the 3D rigid body transformation method in blue and Point Grey camera using the extrinsic camera calibration method in magenta.

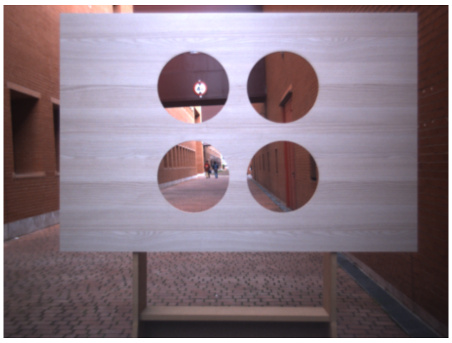


Figure 2.13: The calibration target for method presented in [22].

algorithm to find a final estimation of the extrinsic parameters. The authors concluded that the proposed algorithm outperforms the existing approaches by a large margin.

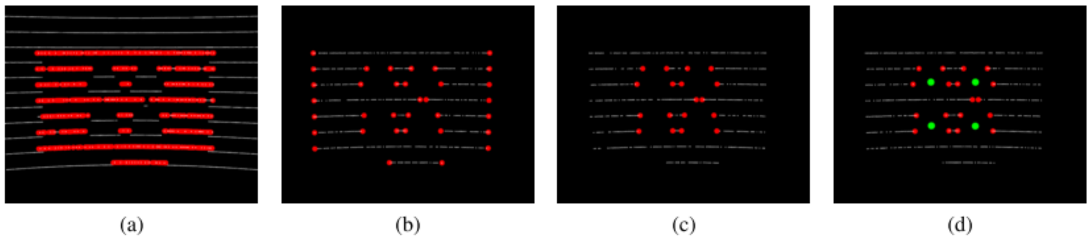


Figure 2.14: Segmentation pipeline for extraction of the reference points from the LRF point cloud [22].

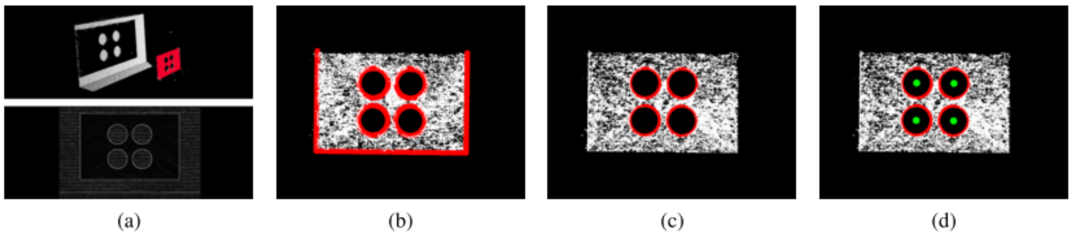


Figure 2.15: Segmentation pipeline for the extraction of the reference points from the stereo point cloud [22].

All previously presented methods, i.e. [28], [51] and [22] propose calibration methodologies which are done in pairs of sensors. In [28], the calibration process requires intervention by the user. In [36], in spite of being a multi-sensor calibration method, it needs specific lighting conditions to work, which implies needing an appropriate place to perform the calibration.

2.3 Camera to camera calibration

Nowadays there are some tools to carry out the calibration of a dual camera system like the Matlab Camera Calibration Toolbox [11] and the algorithm implemented in OpenCV [15]. These tools allow the calibration of two cameras, where a chessboard is used as a reference pattern. However, these methods do not allow us to calibrate N cameras at the same time.

In 2013, Heng et. al. presented a new method to calibrate a multi-camera system [24]. In this work, the calibration method is projected to calibrate a multi-camera system in which each pair of cameras is arranged in a stereo configuration. However, the authors point out that this method can easily be extended to a multi-camera system with at least one stereo camera and any number of monocular cameras. However, they do not refer to how this can be achieved. In this method the whole system, cameras plus the camera support, is in motion during the calibration process.

This method begins with stereo calibration, in which the authors use several chessboards and the method implemented in [34], when the system sees a minimum of chessboards, it finds the initial estimation for the camera pose and intrinsic parameters. Then, having the camera pose, the authors estimate the transformation between two cameras and thus get the stereo configuration.

After stereo calibration, each stereo camera rebuilds the scenario on the respective coordinate system and using the Center Surround Extremas (CenSurE) feature detector [8], implemented in OpenCV, they detect features in each stereo data. Using a P3P method [26], they find correspondences of the features detected by each stereo camera, then using the RANSAC method, they get an initial estimation of the transformation between stereo cameras. In parallel, a graph node is built where the nodes are the stereo camera poses and the edges between nodes correspond to the correspondence features between the stereo cameras. These two processes run at the same time in loop, and finish when the connections between nodes are well supported.

Having a lot of corresponding features between the stereo cameras, the authors merge all stereo maps and by combining the detected features they could estimate the transformations between all stereo systems. After having the initial estimates of the transformations between the stereo cameras, the authors run a bundle adjustment to optimize the camera intrinsics, camera extrinsic parameters, camera poses, and 3D scene points. The bundle adjustment uses a complex cost function (Eq. 2.8) to minimize the reprojection errors, where the first set of residuals corresponds to the sum of image reprojection errors of the 3D scene points, and the second corresponds to the sum of image reprojection errors of the chessboard corner points.

$$\begin{aligned} &CF(K_c, P_i, Q_j, T_c, X_p) = \\ \sum_{c,i,p} &w_p \rho(\|\pi_1(K_c, P_i, T_c, X_p) - p_{cip}\|^2) \\ &+ \sum_{c,j,q} \rho(\|\pi_2(K_c, Q_j, Y_q) - p_{cjq}\|^2), \end{aligned} \tag{2.8}$$

π_1 is a projection function that predicts the image coordinates of the scene point X_p seen in camera c given the camera's intrinsic parameters K_c, the camera system pose P_i, and the transformation from the camera frame to the camera system frame T_c. p_{cip} is the observed image coordinates of X_p seen in camera c with the corresponding camera system pose P_i. Similarly, π_2 is a projection function that predicts the image coordinates of the chessboard corner point Y_q seen in camera c given the camera's intrinsic parameters K_c, and the camera poses Q_j. p_{cjq} is the observed image coordinates of Y_q seen in camera c whose pose is Q_j. ρ is a robust cost function used for minimizing the influence of outliers.

The calibration takes around 15 minutes, and the resulting average reprojection error associated with the generated map was 0.659 pixels. The authors point out that this calibration method produces accurate extrinsic calibration parameters with multiple stereo cameras, with a condition of that there is a sufficient number of inter-camera feature correspondences, and the majority of scene points are close to the cameras.

Most of the methods presented to calibrate a multi-camera system use a pairwise calibration and often use an object with known shape and dimensions or a chessboard. On the other hand, there are other methods like the method proposed in [24] that uses an global extrinsic calibration of the cameras, but before obtaining the extrinsic parameters, the method carries out a stereo configuration between a pair of cameras and then computes the transformations between the stereo cameras and only then does it get the extrinsic parameters of all cameras. This document will present a new method, where the global calibration of N cameras is carried out, without using any of the above methods. The advantages of this new approach are that this method does not require any user intervention, the calibration of N cameras is done at the same time, the cameras do not have any restriction in their position and the process is totally automatic.

2.4 Structure-from-Motion

Structure-from-Motion (SFM) is the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints. This technique was developed in the branches of computer vision and visual perception, and nowadays it has many applications like augmented reality, autonomous navigation, motion capture, hand-eye calibration, image/video processing, image-based 3D modelling, remote sensing, image organization/browsing, segmentation and recognition, and military applications [52]. The SFM process can summarize in a multi-camera calibration, where there are as many cameras as there are images. The calibration of these cameras allows the rebuilding of the 3D scenario existent on images. The SFM have two implemented methods, the Bundle Adjustment (BA) and the Sparse Bundle Adjustment (SBA).

The BA is the problem of refining a visual reconstruction to produce *jointly optimal* 3D structure and viewing parameter (camera pose and/or calibration) estimates. *Optimal* means that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and *jointly* that the solution is simultaneously optimal with respect to both structure and camera variations [50].

The SBA is an identical problem but more robust, because it is based on a dense Cholesky factorization of the reduced camera matrix. It has space complexity that is quadratic and time complexity that is cubic in the number of images [31].

In 2010, Agarwal et. al. presented the design and implementation of a new inexact Newton type BA algorithm [7]. Initially, they have a set of 3D points in the real world. Then they get some images of these points by different cameras, where each camera is defined by its orientation and translation relative to a reference frame and also by focal length and two radial distortion parameters. After this acquisition, the authors project the 3D points into the image and get the 2D coordinates of the points seen by the cameras. The goal of the BA algorithm is to adjust the initial estimation of the camera parameters and the position of the points, in order to minimize the reprojection errors. Mathematically the algorithm works according to the mathematical expression illustrated in Eq. 2.9.

$$\min_{a_j, b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} d(Q(a_j, b_i), x_{ij})^2, \tag{2.9}$$

where each camera j is parametrized by a vector a_j and each 3D point i by a vector b_i and $Q(a_j, b_i)$ is the predicted projection of point i on image j and $d(x, y)$ denotes the Euclidean distance between the image points represented by vectors x and y [4], [23].

After the authors construct Jacobian sparsity structure, which consists of a sparse matrix where the number of the lines (M) is the number of point correspondences on the image and the number of the columns (N) is the number of parameters that will be

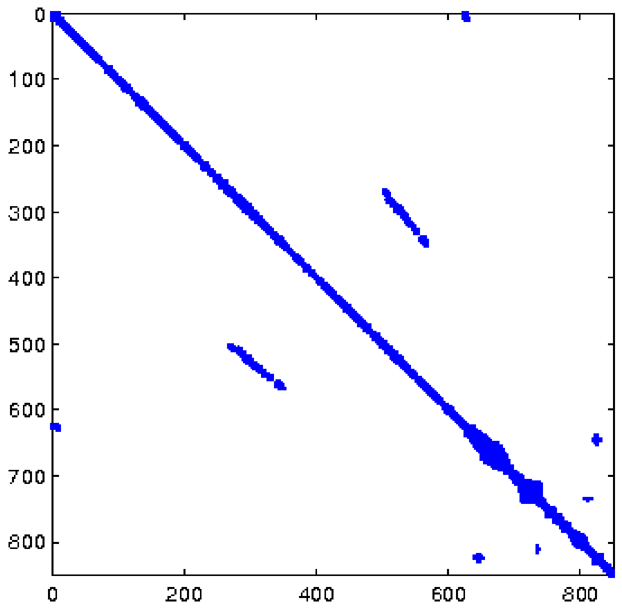


Figure 2.16: An example of the sparse matrix used to solve a bundle adjustment problem. [27].

optimized. The matrix is constituted of zeros and ones where the ones are collocated in the line matrix (m) that corresponds to a point correspondence when it is affected by the parameter (n), an example of a sparse matrix is illustrated in Fig. 2.16.

Before carrying out the optimization, the authors make a cost function which seeks to minimize the Euclidean distance (Eq. 2.2) between the points of the image and the correspondent points projected in the image. Finally, having an initial estimation, a cost function and a sparse matrix, the authors give it as input on the optimization function and they get the optimal parameters values as output that results in a minimal reprojection error.

Chapter 3

Experimental infrastructure

The hardware and software used in this work are presented in this chapter, as well as its characteristics and the method of use. Although the calibration method presented in this work is able to calibrate N cameras, only one camera sensor was used, where several images were acquired simulating N cameras. The framework and libraries used in this work were the Robot Operating System (ROS), OpenCV and Scientific computing in Python (SciPy). ROS is used to connect to the camera and collect the images. The OpenCV library is used to get the intrinsic parameters of the camera and the distortions coefficients and, at a later stage, to detect the Aruco markers. The SciPy library is used to form the final optimization of the position of the cameras. Next, the proprieties of the software and hardware are presented.

3.1 Software

This section presents all the software used to perform this work, framework and libraries. Its general applications, the method of use and the functionalities used in this work will also be presented.

3.1.1 ROS

ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms [44]. ROS provides services of an operating system such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message-passing between processes, and package management.

The architecture of the ROS processes is represented on a graph architecture, where the processes are carried out on nodes, which can send and receive messages like post and multiplex sensor, control, state, planning, actuator and others. ROS has other packages in which commonly used functionalities are implemented, as well as hardware drivers, robot models, datatypes, planning, perception, simultaneous localization and mapping, simulation tools, and other algorithms.

In this work, the ROS framework was used to connect to the camera and acquire images. For this, a package of hardware drivers of the camera [43] is used, which is composed of just one node for a single camera, that publish two topics, one with the unprocessed image data and the other which contains the camera calibration. The second topic is not used because the calibration of the camera is done posteriorly. To visualize the camera output the *rviz* interface was used which allows saving the images obtained by the camera.

3.1.2 OpenCV

OpenCV is an open-source, computer-vision library for extracting and processing meaningful data from images [12]. This library was designed for computational efficiency and with a strong focus on real-time applications. It contains many algorithms based on the literature. Some examples are object detection, recognizing all or parts of objects, algorithms to get the intrinsic and extrinsic parameters of a camera, tracking objects in motion and others. OpenCV has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. In order to have the library organised, OpenCV is divided in modules, the *calib3d* and *aruco* modules are used in this work.

Module *calib3d*

This module has many functions to get the calibration parameters of the camera and it also has functions to rebuild the 3D scenario using 2D images. The functions to obtain the intrinsic parameters of the camera as well as the lens distortions were used in this work, where the functions only need some images obtained by the camera. These images must have a chessboard and it is necessary to give the real dimensions of the squares of the chessboard.

Module *aruco*

This module is an extra module of the OpenCV, and it has functions to detect Aruco markers in an image and the corners of the marker. Having these markers in the image, and knowing the real size of the markers, it is possible to obtain the camera pose relative to the marker on the 3D space by using the functions of this module.

3.1.3 SciPy

SciPy is an open source library which contains routines commonly used in scientific work. There are routines for computing integrals numerically, solving differential equations, optimization and sparse matrices. This library is implemented in python programming language. A routine for an optimization problem is used in this work. The optimization problem of this work is a Large-scale bundle adjustment problem that follows the algorithm presented in [7].

3.2 Hardware

This section presents all the hardware used to carry out this work, as well as its characteristics and utilities. In this work only two pieces of equipment were used. First a point gray camera was used for image acquisition. In an advanced phase, an ASUS ZenFone AR was used, which contains a 3D camera that allows the reconstruction of spaces.

3.2.1 Point Grey Camera

The camera that was used in this work was the Point Grey Flea3 FL3-GE-28S4 Camera (Fig. 3.1) which is a camera that promotes high-performance for industrial environments and traffic applications. This camera collects high resolution and quality images. Combining these characteristics with the sensitivity and size of the camera makes this model ideal for applications in factory automation and machine vision. Some of the most relevant characteristics of the camera are represented in the Tab. 3.1.

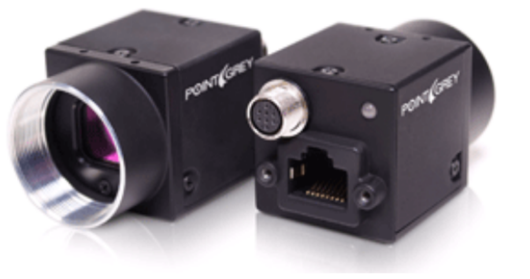


Figure 3.1: Point Grey camera FL3-GE-28S4-C [3].

Resolution	1928×1448
Framerate	15 FPS
Megapixels	2.8 MP
Chroma	Color
Pixel Size	3.69 μm
Interface	GigE Vision
Power Requirements	12-24 V
Dimensions	29mm×29mm×30mm

Table 3.1: Proprieties of the Point Grey camera FL3-GE-28S4-C [42]

3.2.2 ASUS ZenFone AR

The ASUS ZenFone AR is a device that supports Google platforms dedicated to emerging technologies and it is available to support both Tango and Daydream. Project Tango is Google's attempt to get mobile phones and tablets to see the way that the humans see. This means granting the device full spatial awareness, or the ability to understand your environment and your relation to it. The essential aim is to allow smartphones to understand the world around them, enabling them to provide augmented reality experiences. Daydream is a virtual reality platform developed by Google that is built into the Android mobile operating system.

To enable this technology, the smartphone is equipped with some sensors, as shown in Fig. 3.2. The built-in sensors are: main shooter 23 MP IMX 318, dual-tone LED flash with RGB color correction sensor, one laser autofocus, one depth sensing camera and one motion tracking camera.

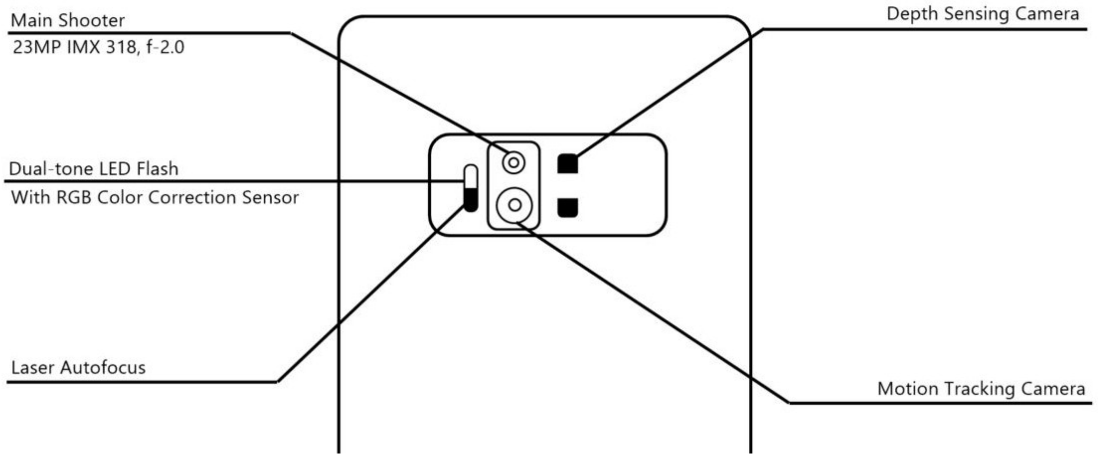


Figure 3.2: ASUS ZenFone AR sensors setup (rear camera (s)).

With all these sensors and using a previously developed treatment application, it is possible to create a 3D point cloud of the surrounding space. Subsequently a triangular mesh is defined which is painted using the colour mapping resulting from the treatment of the various images obtained with the camera, combined with the data obtained from the other sensors.

Chapter 4

Calibration approach

This chapter presents the methodology for calibrating several cameras. As was mentioned in chapter 2, the main purpose of calibrating a camera is to obtain the extrinsic parameters with respect to a common reference frame. Thus, the final result of the calibration process is one transformation matrix, as represented on Eq. 4.1, for each camera.

The method presented in this work is an automatic calibration method which calibrates all the cameras at the same time. This method has no restrictions on the initial poses of the cameras. However, for this method to work, the camera must have at least one connection marker with another camera. To manage and process the connectivity between cameras, a graph-based approach is used. This calibration method assumes that the intrinsic calibration of the cameras has already been done, and therefore the cameras' intrinsic parameters are known.

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.1 Intrinsic camera calibration

The intrinsic calibration of the camera is done only once for each camera sensor, it assumes that the camera has no adjustable focus. As was said above, it is necessary to obtain these parameters before doing the extrinsic calibration. In subsection 2.2, what intrinsic calibration is and what the camera's intrinsic parameters are, were explained. This process is done with an algorithm implemented in OpenCV [15]. The OpenCV algorithm uses some pictures with a chessboard of known (Fig. 4.1) dimensions. In order to improve the calibration results, the chessboard should vary the rotation around the *Pitch* axis and *Yaw* axis and it is also important to move the chessboard through the entire image size.

When a chessboard is used to calibrate a camera, the top left corner point is defined as the origin of the global reference system, as illustrated, in red, in Fig. 4.2. The function implemented in OpenCV to extract the intrinsic parameters of the camera is the *calibrateCamera* function. This function needs the corner points defined in the global coordinate system and it also needs the detected chessboard corners in the image. The global coordinates of the corner points are calculated using the chessboard size (in this case the chessboard size is 8×6 squares) and the size of the square (in this case 105 mm). To detect the corners of the chessboard the *findChessboardCorners* function is used, which returns the coordinates on the image of all corners of the chessboard, as represented in Fig. 4.2.

Finally, the intrinsic parameters are saved on a file that will be read at the beginning of the extrinsic calibration, at a later stage.

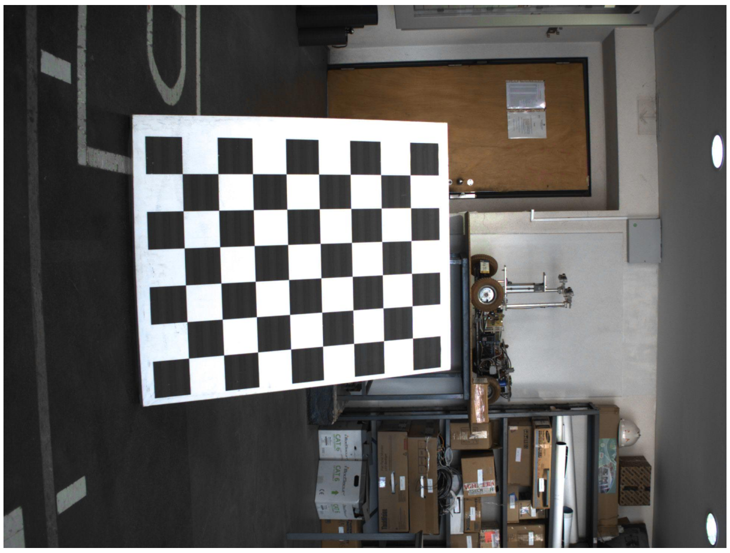


Figure 4.1: Example of an image with a chessboard.

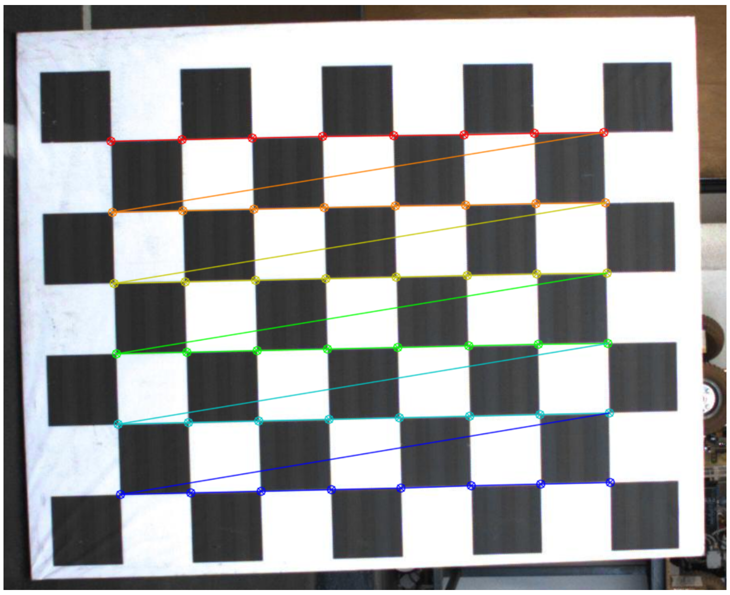


Figure 4.2: Representation of the corners detected by the *findChessboardCorners* function.

4.2 Calibration target

The calibration target used in this work is not just one, but many Aruco markers (Fig. 4.3). These markers are defined in the Aruco library, presented in the *aruco* module of the OpenCV. This is a popular library for the detection of square fiducial markers [20].

Figure 4.3 illustrates that an Aruco marker is a square which has a wide black border to allow the detection of the marker on the image. In the middle of the square, the markers have a binary code which corresponds to an identifier number.

In the *aruco* module, there are many different dictionaries of Arucos which are distinguished by the number of markers that compose the dictionary and the size of the marker (number of bits) [35]. The dictionary used in this work is the dictionary with 1024 markers and a size of 5×5 (25 bits). This dictionary is called "DICT_ARUCO_ORIGINAL" and is present on the *aruco* module. For generating and printing these markers an automatic generator present on the web [1] was used.

The process of detecting Aruco markers on the image begins by detecting squares on the image. Then, applying some filters, the squares that correspond to the markers are

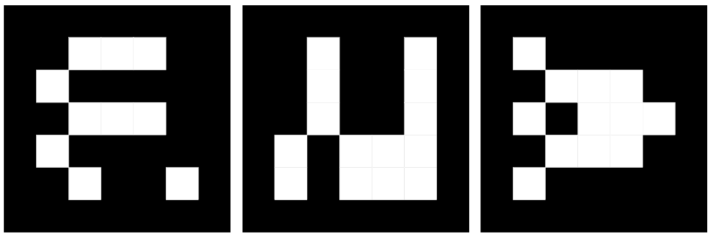


Figure 4.3: Example of markers images.

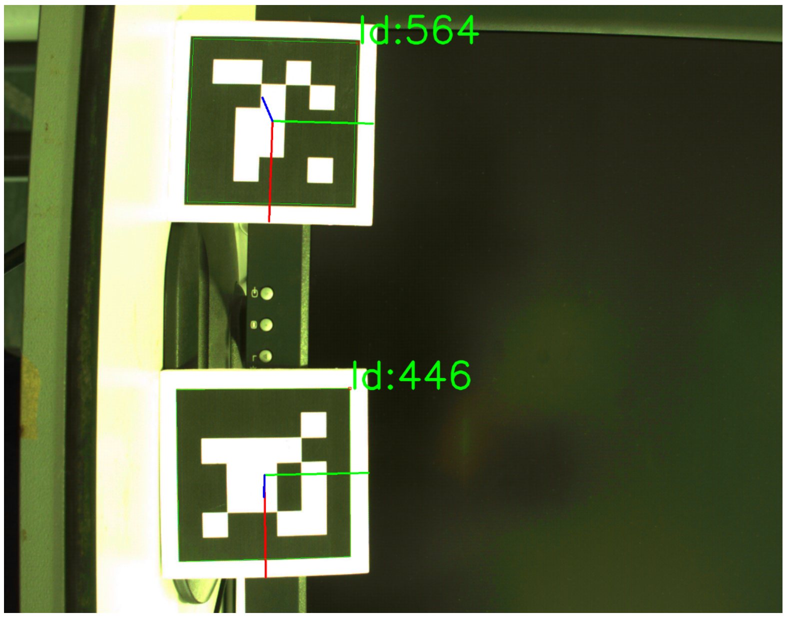


Figure 4.4: Example of markers detection on the image.

detected. Finally, the marker corners and the respective ids are identified. This process is done using the *detectMarkers* function of the *aruco* module. Fig. 4.4 illustrates an example of the marker detection. Each Aruco marker has its own reference system where the x and y axes lie on the marker plane and the z axis is perpendicular to the marker plane and points out the marker. To detect this reference system in the image the orientation of the marker's binary code is used. This enables estimating the rotation and translation from the coordinate system of the marker to the coordinate system of the camera. For this the *estimatePoseSingleMarkers* function of the *aruco* module is used.

4.3 Calibration procedure

The calibration method implemented in this work uses the following steps illustrated in the diagram of Fig. 4.5.

The first step is to place the cameras in the position where the calibration will be done and to spread markers in front of the cameras. Then, it is necessary to obtain one image for each camera. Next, follows for the second step where the markers are detected on the image. This step was explained in section 4.2, in which the *aruco* module is used from the OpenCV library. When a marker is detected a new detection is saved with the id of the marker and the camera by which the marker was seen are saved. In this detection, the transformation from the marker coordinate system to the camera coordinate system where the marker is seen is also saved.

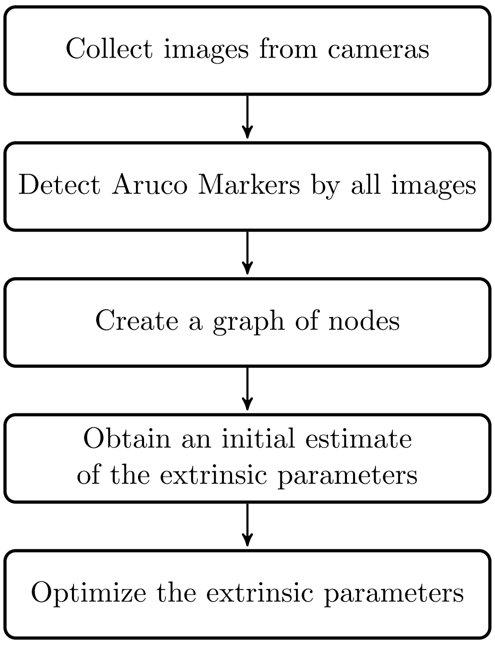

Collect images from cameras
Detect Aruco Markers by all images
Create a graph of nodes
Obtain an initial estimate of the extrinsic parameters
Optimize the extrinsic parameters

Figure 4.5: Overview of the steps of the calibration process.

4.3.1 Create a graph of nodes

The graph node is the way to verify how the problem is connected and it gives a better perception of the problem. The NetworkX [13], which is a Python package, was used. This package, which allows the creation, manipulation, and study of the structure of the graph nodes, was used to construct this graph. After the construction of the graph node, this tool can show if all nodes are connected, which is an important function in this work because if the graph node is connected, this means that all the cameras have at least one Aruco marker that they connect with another camera.

The process of the construction of the graph is represented in Algorithm 1, where, for each detection, two nodes are created, one for the camera and the other for the marker if they do not already exist. Finally, a connection between these nodes is created. The nodes created are cameras and markers, and the connection represents a transformation matrix from the marker to the camera.

Algorithm 1 Constructing a graph node

```
1: for detection in detections do
2:     if the node with the ID of this marker does not exist then
3:         Add a node with the ID of the marker
4:     end if
5:     if the node with the ID of this camera does not exist then
6:         Add a node with the ID of the camera
7:     end if
8:     Add an edge between these nodes
9: end for
```

Fig. 4.6 exemplifies the result of constructing a graph of nodes. In this example, 3 cameras (C) and 3 markers (A) were used, where the cameras "C0" and "C1" are connected by the marker "A595" and the cameras "C1" and "C2" are connected by the marker "A444".

4.3.2 Obtaining an initial estimate

It is necessary to calculate an initial estimate of the extrinsic parameters to give to the optimizer later, since the optimizer needs an initial approximate and coherent estimate of the extrinsic parameters.

The detection of the Aruco markers allows having the transformation from the coordinate system of the marker to the coordinate system of the camera, as is illustrated in Fig. 4.7. However, the extrinsic calibration consists of having the position of the cameras and markers with respect to a common reference system. The graph is used to obtain the initial estimates of transformation from each camera and marker to the map reference system.

At the beginning, before computing the initial estimation for the extrinsic parameters, it is necessary to define the reference frame. The world reference frame can be a marker or a camera, however, the reference frame must be a camera because it is a camera calibration, and the goal of the camera calibration consists of having the positions of the cameras in relation to a world reference frame. Thus, the final results will be the transformations of the cameras and the markers in relation to the camera that was previously selected.

After defining the world reference frame, the conditions for computing an initial estimate for the extrinsic parameters are met. Thus, it is now necessary to find the transformation between each marker or camera and the world reference frame. The path from any node (camera or marker) to the world reference frame is retrieved by the graph. For example, in the graph represented in Fig. 4.6, if "C0" is defined as the world reference frame, the shortest way to go from "C2" to the world reference frame will be "C2→A444→C1→A595→C0". The above problem is simple. However, when a more complex problem is presented, it is possible that there are several possible paths to go from a node to the world reference frame, a function *all_shortest_path* in NetworkX is used to find all shorter paths of the set of paths. Each of these paths has an associated

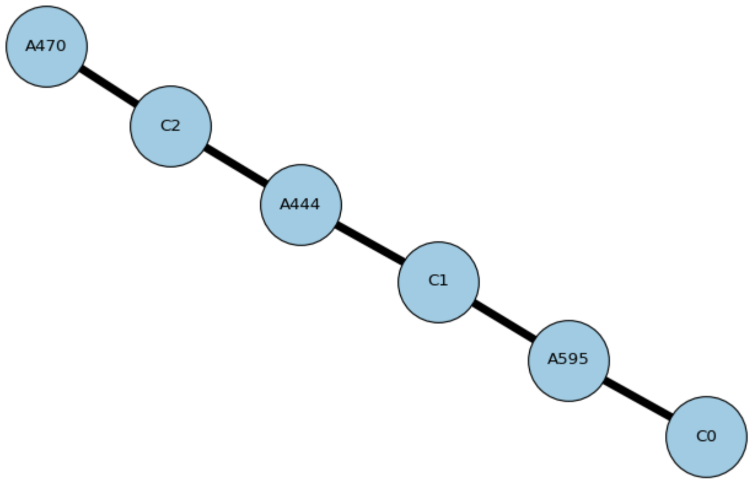


Figure 4.6: Example of a graph node construction.

$$\begin{tikzcd}
\text{Aruco} \arrow[rr, bend left, "{}^{A}T_{C}"] & & \text{Camera} \arrow[ll, bend left, "{}^{A}T_{C}^{-1}"]
\end{tikzcd}$$

Figure 4.7: Diagram of the transformation from the coordinate system of the marker to the coordinate system of the camera.

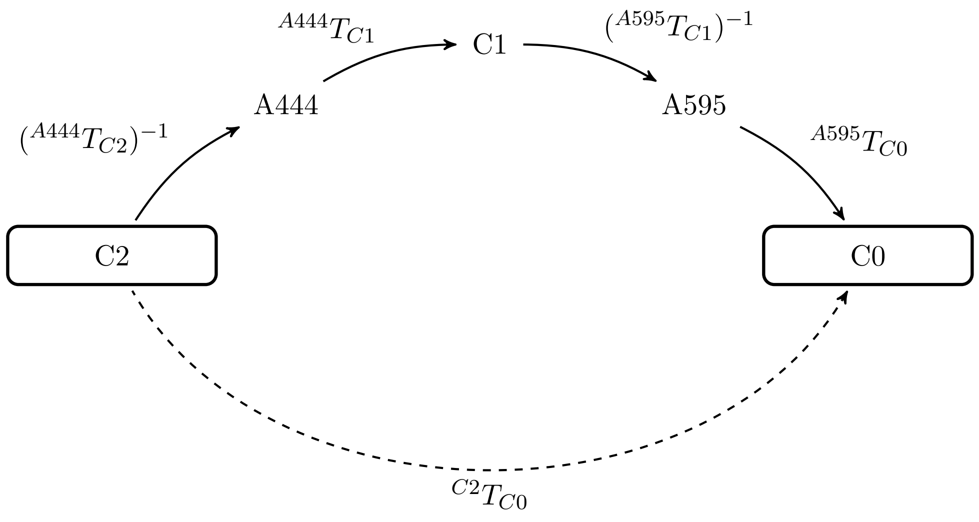

$$\begin{tikzcd}
 & A444 \arrow[r, bend left, "{}^{A444}T_{C1}"] & C1 \arrow[r, bend left, "({}^{A595}T_{C1})^{-1}"] & A595 \arrow[dr, bend left, "{}^{A595}T_{C0}"] & \\
\boxed{C2} \arrow[ur, bend left, "({}^{A444}T_{C2})^{-1}"] \arrow[rrrr, dashed, bend right, "{}^{C2}T_{C0}"'] & & & & \boxed{C0}
\end{tikzcd}$$

Figure 4.8: Diagram of the transformation from the "C2" node to the "C0" node.

transformation. As the goal is to obtain a single transformation, the average of the various transformations obtained is computed.

Compute the average of multiple transformations is not a linear process. Therefore, to compute the average transformation, we begin by computing the average of all translations (x, y and z). Then, the average of the rotation component has to be computed. Initially, the rotation component is converted to rotation quaternions [5]. Next, the spherical linear interpolation between two rotations is carried out [6], which corresponds to the average of the two rotation components. By crossing all rotation components and using the previously computed average, the process of computing the spherical linear interpolation is repeated until the average of all rotation components is computed. Finally, both the rotation and translation components are linked, constructing the average transformation matrix. Thus, using the transformations obtained from the detection of the markers, it is possible to estimate the transformations of all markers and cameras in relation to the world reference frame.

The diagram represented in Fig. 4.8 explains the example previously refereed to. For example, in this case camera "C2" only sees marker "A444", so in this detection it is possible to know the transformation from the coordinate system of the marker to the coordinate system of the camera. However, the required transformation is the transformation from the camera to the marker, so the inverse of the transformation of the marker to the camera is used. This is given by:

$$ {}^{a}T_{b} = ({}^{b}T_{a})^{-1} \tag{4.2}$$

From diagram in Fig. 4.8, we can extract Eq. 4.3 which represents the initial estimate of the extrinsic calibration of camera "C2".

$$ {}^{C2}T_{C0} = ({}^{A444}T_{C2})^{-1} \cdot {}^{A444}T_{C1} \cdot ({}^{A595}T_{C1})^{-1} \cdot {}^{A595}T_{C0} \tag{4.3}$$

The algorithm which performs this process is represented in Algorithm 2.

Algorithm 2 Getting an initial estimate of the extrinsic parameters.

```
1: for node in nodes do
2:     find all the shortest paths from the node to the reference node
3:     for path in paths do
4:         T = identity matrix
5:         for i in [1,2,.., until the number of nodes that constitute the path] do
6:             start_node = path[i-1]
7:             end_node = path[i]
8:             det = detection which has the marker and the camera that represents the start and end node
9:             if start_node is a marker then
10:                Ti = transformation that corresponds to the detection
11:            else
12:                Ti = inverse of the transformation that corresponds to the detection
13:            end if
14:            T = Ti × T
15:        end for
16:        Join T to AllTransformations list
17:    end for
18:    $T_{med}$ = mean of AllTransformations list
19:    if node is a camera then
20:        $T_{med}$ corresponds to the initial estimate of the extrinsic calibration of the correspondent camera
21:    else
22:        $T_{med}$ corresponds to the initial estimate of the extrinsic calibration of the correspondent marker
23:    end if
24: end for
```

4.3.3 Optimization of the extrinsic parameters

The optimization was done based on work developed by Agarwal [7] which is referred to in subsection 2.4. In work previously referred to, the authors project the 3D points onto the image and compare the distance between the projected points and the points detected by image processing.

Creating the vector that will be optimized

To perform the optimization, it is necessary to create a vector of parameters with all the parameters that will be optimized. The parameters that will be optimized are the transformation matrices of the cameras and the markers. The vector will have the format represented in Eq. 4.4.

$$\text{vector} = [{t_x}^{C0}, {t_y}^{C0}, {t_z}^{C0}, {r_1}^{C0}, {r_2}^{C0}, {r_3}^{C0}, ..., {t_x}^{A0}, {t_y}^{A0}, {t_z}^{A0}, {r_1}^{A0}, {r_2}^{A0}, {r_3}^{A0}, ...] \quad (4.4)$$

t_x, t_y, t_z represent the translation component and r_1, r_2, r_3 represent the rotation component, where the rotation matrix component is converted on Rodrigues' rotation so that there is no redundancy in the rotation component. These 6 elements of the vector represent just one transform matrix whereby, the number of elements of the vector will be equal to $6 \cdot (n_{\text{cameras}} + n_{\text{markers}})$.

Projecting 3D points onto 2D points on the image

In this work the same process refereed to above is used, where the 3D points that represent the corners of the Aruco marker are projected onto the image, as illustrated with blue circles in Fig. 4.9. The transformation of the 3D points onto 2D image points made in this work follows the pinhole camera model [19]. To transform the 3D points into 2D image points the distortion coefficients, the intrinsic matrix and the extrinsic matrix are necessary.

Initially, the corners of the markers are transformed from the coordinate system of the marker to the coordinate system of the world reference frame, using Eq. 4.5, where T represents the extrinsic matrix. The 3D points represented in the marker coordinate system are calculated using the size of the Aruco marker. For example, if the size of the marker is "s", coordinates of the corners are $(\frac{-s}{2};\frac{s}{2})$, $(\frac{s}{2};\frac{s}{2})$, $(\frac{s}{2};\frac{-s}{2})$ and $(\frac{-s}{2};\frac{-s}{2})$, beginning in the upper left corner and rotating clockwise, respectively.

$$P_{\text{reference frame}} = T \cdot P_{\text{marker}} \quad (4.5)$$

The coordinates of the $P_{\text{reference frame}}$ can be represented by (x, y, z). After getting the points coordinates on the reference frame, the 3D coordinates are converted into homogeneous coordinates with the Eq. 4.6.

$$\begin{aligned} x' &= \frac{x}{z} \\ y' &= \frac{y}{z} \end{aligned} \quad (4.6)$$

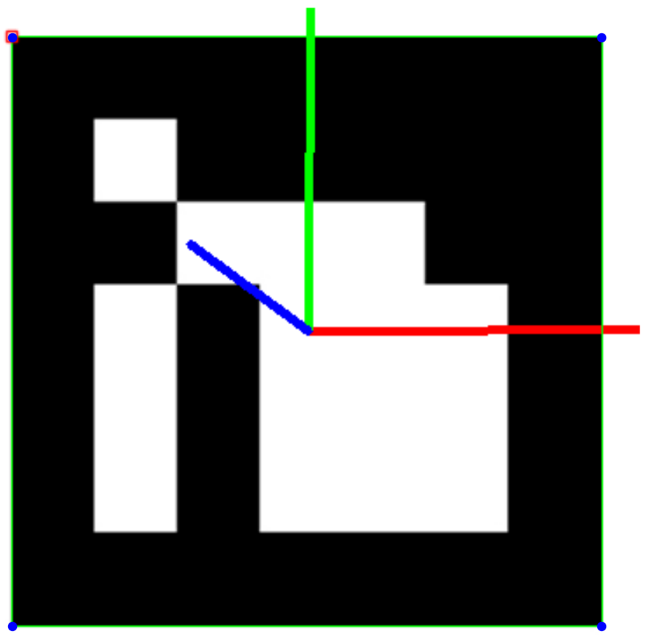


Figure 4.9: Representation of the corners of the Aruco marker.

Then, the tangential and radial distortion are removed, using Eqs. 4.7, which results from Eq. 2.3 and 2.4 refereed to in subsection 2.2.

$$\begin{aligned} x'' &= x' \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + 2 \cdot p_1 \cdot x' \cdot y' + p_2 \cdot (r^2 + 2 \cdot x'^2) \\ y'' &= y' \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + p_1 \cdot (r^2 + 2 \cdot y'^2) + 2 \cdot p_2 \cdot x' \cdot y' \\ &\text{where } r = \sqrt{x'^2 + y'^2} \end{aligned} \tag{4.7}$$

Finally, using the intrinsic matrix (Eq. 2.5), that results in Eq. 4.8, the coordinates of the points in the image (u, v) are obtained.

$$\begin{aligned} u &= f_x \cdot x'' + c_x \\ v &= f_y \cdot y'' + c_y \end{aligned} \tag{4.8}$$

Sparse matrix

The optimization of a large problem, i.e. one with thousands of parameters, may become cumbersome. This is due to the difficulty in tuning a very large number of parameters from the scalar number returned by the cost function. In multi-camera calibration problems the relationships between the various cameras is not complete. In other words, if one considers a graph where the cameras and aruco markers are the nodes and transformations between these are edges, this graph will not be fully connected. This demonstrates the fact that not all cameras observe all of the aruco markers.

Taking this into account one may change the cost function to return a vector of errors instead of a scalar error. Each value in this vector will represent the reprojection error associated with each camera-aruco marker detection. Since it is possible to define which optimization parameters affect a camera-aruco marker detection, i.e., a camera X to aruco marker Y detection is influenced only by the six parameters that describe the pose of the camera and the six parameters that describe the pose of the aruco marker. A matrix of detections in the Y axis and optimization parameter in the X axis is created to describe which detections (i.e. positions in the error vector returned by the cost function) are influenced by which parameters. Since, in multi-camera optimization problems, this matrix is populated by a large number of zeros (non influences), this is often called sparse matrix and the problem sparse bundle adjustment.

By using the sparse matrix information, the optimizer is able to tune the optimization parameters more precisely and thus the optimization is much faster. In some cases involving very large problems, a sparse formulation is the only way to achieve a solution since the classical approach fails to converge to a solution.

For the example of the node graph in Fig. 4.6, the sparse matrix would look like it is represented in Tab. 4.1, where, in the columns, there are the extrinsic parameters corresponding to each element (camera or marker) which are included in the vector to optimize, and the lines correspond to detections. Each element of the columns corresponds to six elements, the first three correspond to the translation component of the transformation matrix and the following correspond to the rotation component, the component of rotation is represented in Rodrigues angles (tx,ty,tz,r1,r2,r3). Therefore, each element of the line correspond to six equal elements and the sparse matrix of this example have a size of 5×36. The ones of sparse matrix mean that the detection has influence on the parameters, and the zeros mean the opposite.

Detections/Parameters	C0	C1	C2	A444	A470	A595
C0 - A595	1	0	0	0	0	1
C1 - A595	0	1	0	0	0	1
C1 - A444	0	1	0	1	0	0
C2 - A444	0	0	1	1	0	0
C2 - A470	0	0	1	0	1	0

Table 4.1: Representation of the sparse matrix corresponding to the example in Fig. 4.6

Setting the bounds for the parameters

A bundle adjustment problem can be quite a time-consuming process due to the existence of several solutions, but sometimes most of the existing solutions are not correct. For this, it is possible to set limits for the vector parameters to optimize, so that the optimizer can reduce the number of possible solutions and converge to a solution closer to the desired values.

In this work, this functionality was only used in the tests of dataset 4, as it was intended to find the positions of all the centers of the markers in relation to the marker "A0". So, it was necessary to tell the optimizer that the transformation associated with the marker "A0" would have to be maintained, to avoid redundancies. In this way it was possible to converge the optimizer to the solution that was intended.

Cost function

The optimization process aims to decrease the reprojection error. The reprojection error is the Euclidean distance between the point reprojected in the image and the interest point of the image detected by image treatment (ground truth). Thus, the cost function used in this work is a cost function that seeks to minimize the Euclidean distance between the points reprojected in the image and the points detected by image processing in the image of the markers' corners.

The cost function is defined in Eq. 4.9 where N_d is the number of detections of the marker/camera pairs, N_c is the number of markers' corners and $x_{\text{reprojection}}$, $y_{\text{reprojection}}$ and x_{img}, y_{img}, as the name indicates, are respectively, the coordinates x, y of the reprojection of the 3D points and the coordinates of the points detected in the image.

$$\text{Cost function} = \sum_{i=1}^{N_d} \sum_{j=1}^{N_c} \sqrt{(x^{ij}_{\text{reprojection}} - x^{ij}_{\text{img}})^2 + (y^{ij}_{\text{reprojection}} - y^{ij}_{\text{img}})^2} \tag{4.9}$$

Another cost function (Eq. 4.10) was tested where the Euclidean distance between the reprojected points in the image and the points detected by image processing in the image of the marker centers is minimized. This function is, in all, equal to the cost function referred to above, with a small difference that for each marker only the center point is compared instead of the corners.

$$\text{Cost function} = \sum_{i=1}^{N_d} \sqrt{(x^{i}_{\text{reprojection}} - x^{i}_{\text{img}})^2 + (y^{i}_{\text{reprojection}} - y^{i}_{\text{img}})^2} \tag{4.10}$$

Optimization algorithm

The optimization is performed like a nonlinear least-squares problem with bounds on the variables, for this the SciPy library is used. This method uses the Trust Region Reflective (trf) algorithm, which is particularly suitable for large sparse problems with bounds.

The SciPy function named "least_squares", only needs the cost function, the vector, the bounds and the sparse matrix to carry out the optimization. At the end of the optimization, the function returns a vector with better results for the extrinsic parameters which minimize the reprojection errors.

Chapter 5

Experiments and results

In order to test and validate the calibration method implemented in this work, some datasets have been developed from the simplest to the most complex. These datasets will be presented in this chapter as well as the results obtained in each.

This calibration method is one that can calibrate several cameras at the same time, so it will be evaluated whether the final result is acceptable not only for the calibration of one camera but also for that of multiple cameras. The number of markers used in the calibration of the cameras should have an influence on the calibration result, and therefore will also be tested.

A new dataset will also be tested in which the poses of the markers are known i.e. there is ground truth, in order to have a numerical validation of the method. Finally, the application of this method will be evaluated in an application of 3D reconstruction of a scene.

This chapter presents the datasets created to test the calibration method that was developed in this work. An explanation of the calibration method in each dataset will also be given. Finally, the calibration results obtained for each dataset will be presented.

5.1 Dataset 1

Initially, a very simple dataset was tested with only one camera and two markers. This simple dataset is designed to validate the calibration process. Fig. 5.1 shows the image obtained from the camera where two markers are visible.

5.1.1 Calibration using the marker's corners

Initially, markers are detected in the images. Then, for each detected marker, the points of the corners (red squares), the identification number (id) referent to each marker and the respective coordinate system of each marker whose position and orientation, as previously stated, depends on the binary code of the marker are represented. As already mentioned, the detection of an Aruco marker allowed us to know the dimensions of the markers, in order to compute the transformation from the camera to the marker.

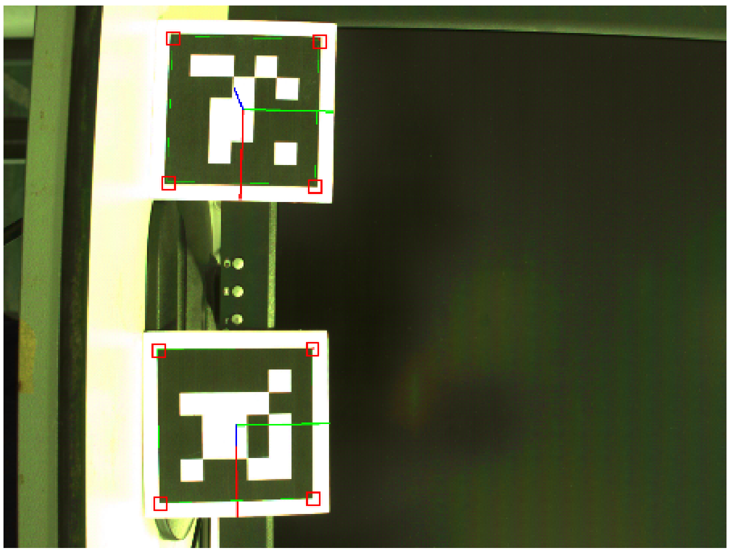


Figure 5.1: Representation of the dataset 1 and markers' corners.

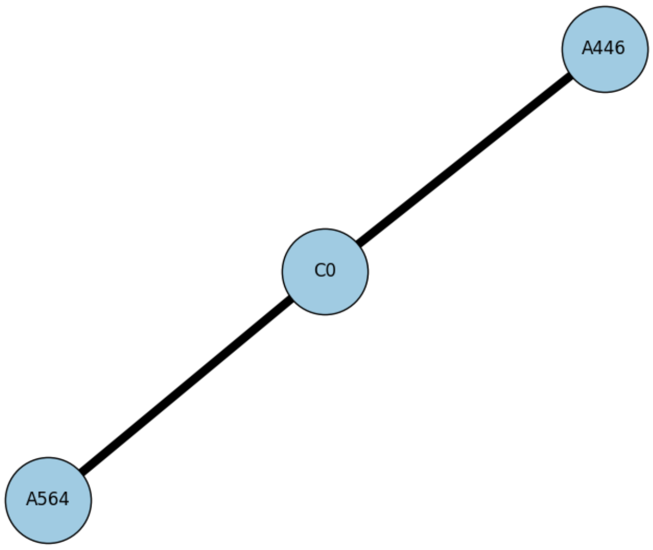


Figure 5.2: Graph node of the dataset 1.

Figure 5.2 represents the graph of nodes referring to dataset 1 where it is visible that the two markers are connected to the same camera. As all the elements of the graph are connected, it is possible to estimate the extrinsic parameters of all elements with respect to the common reference frame. In this dataset, camera 0 is chosen to be the world reference frame. This dataset does not refer to a multi-camera calibration. However, it helps to see if the position of the markers with respect to the camera is consistent. In this way, it is proven that estimating the position of the camera relative to the markers using the Aruco markers is a valid procedure.

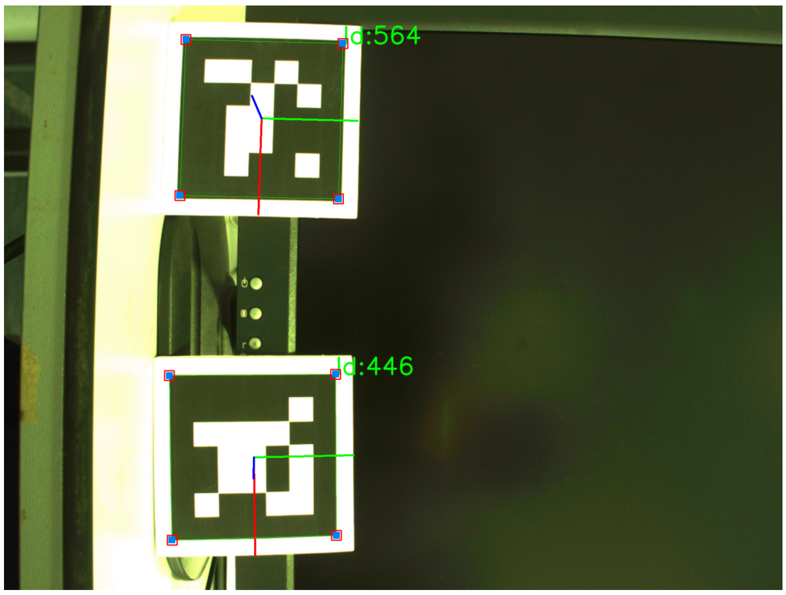


Figure 5.3: Representation of the initial guess (blue squares), the ground truth (red squares) and the id of the marker.

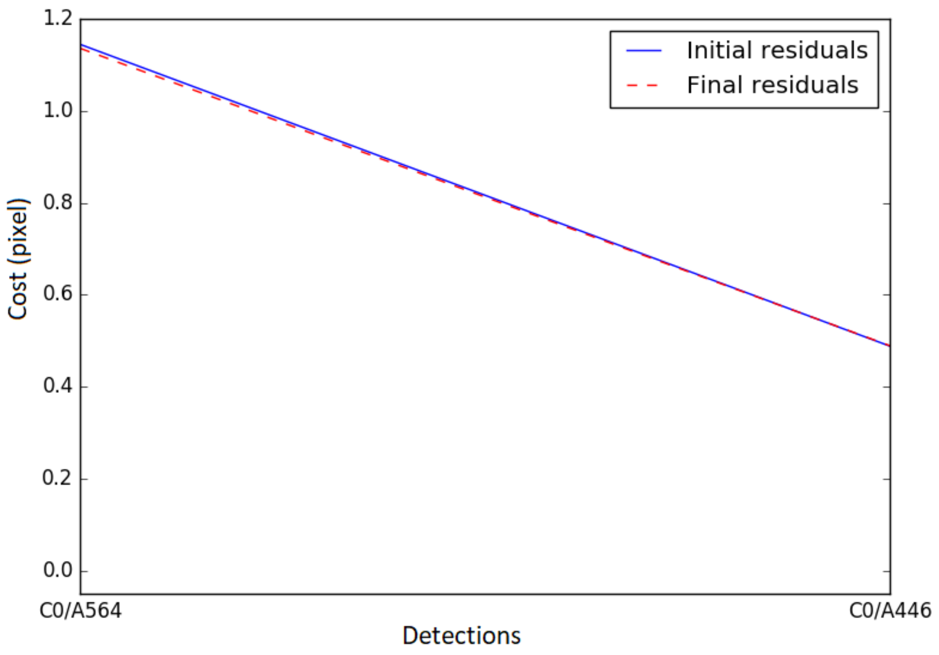


Figure 5.4: Representation of the costs related to each detection.

Projecting the corners of the markers from their coordinates in the global coordinate system to the coordinates in the image results in the blue squares is shown in Fig. 5.3. From this projection it is possible to conclude that the initial estimate is already good. In other words, the reprojection error is already very close to zero.

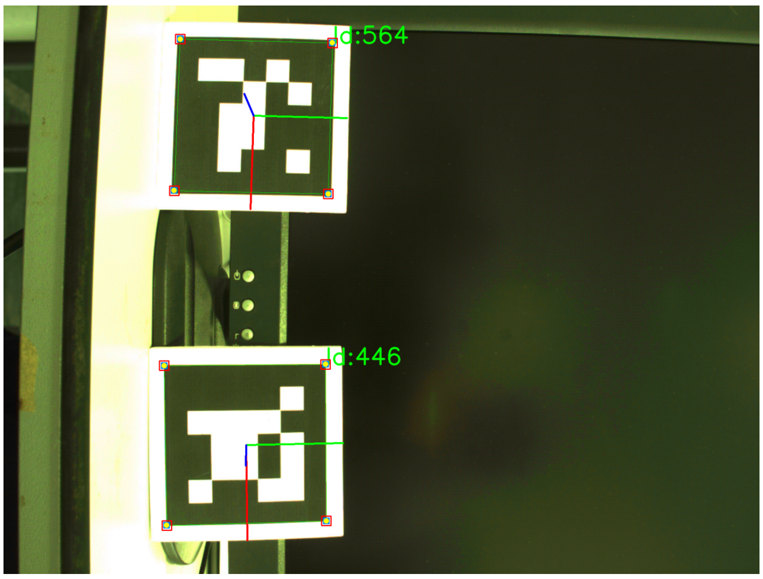


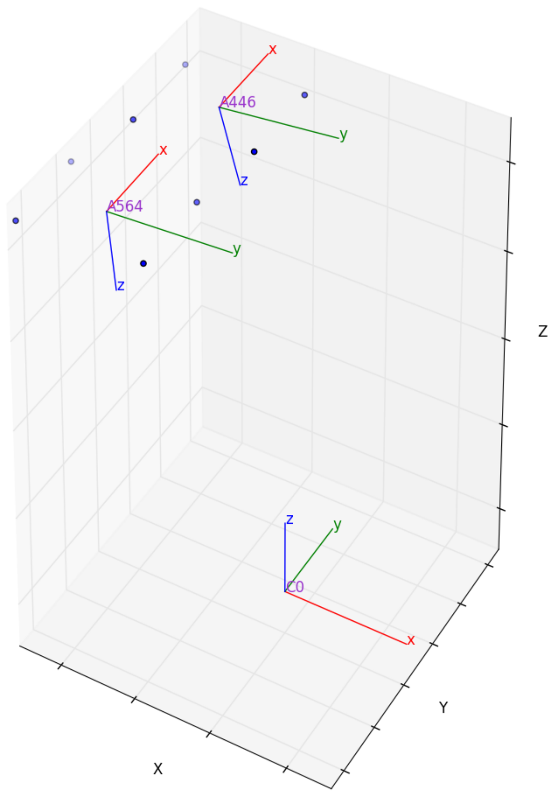
Figure 5.5: Reprojection of the 3D points on the image using the final values resulting from the optimization.

An optimization of the reprojection error is carried out. The results are shown in Fig. 5.4. The graph shows the cost for each of the detections. As mentioned in the previous chapter, the detections represent a camera-marker pair. Therefore, in the case of this dataset, there are two detections that are the pair C0-A564 and C0-A446, where C0 represents the camera of the dataset and the A564 and A446 are the markers present in the dataset.

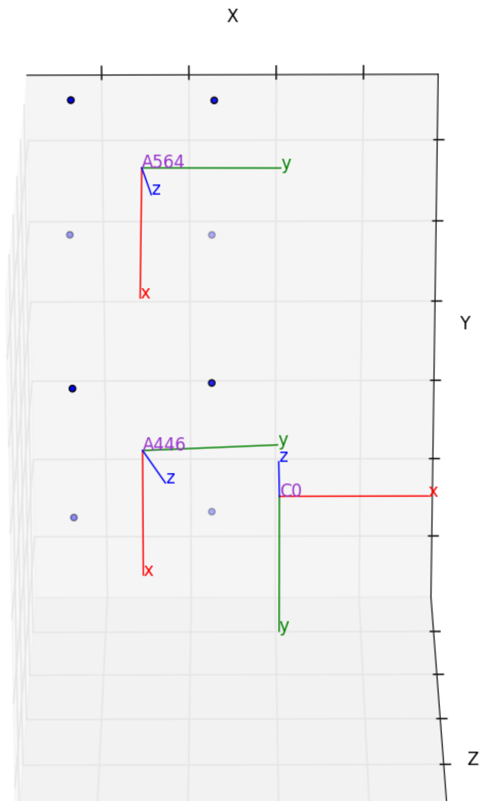
In Fig. 5.4, the solid line represents the initial cost before the optimization and the dashed line represents the result after optimization. From this data we conclude that there was an improvement from the initial solution, with the optimizer achieving more improvements in the cost of the first detection, which initially had a higher cost. In conclusion, the cost of the first and the second detection did not show a very significant variation with the optimization, indicating that the initial estimate was very close to reality. However, the average reprojection error reduced from 0.816 to 0.812 pixels.

In Fig. 5.5 the 3D points are reprojected in the image, using the results obtained by the optimization (yellow circles). The variation from the initial state is almost invisible because the initial estimate already produced very appreciable results.

Thus, obtaining the extrinsic parameters, it is possible to perform a 3D reconstruction of the scenario of dataset 1. This is represented in Fig. 5.6. From this 3D representation it can be concluded that the calibration presents excellent results. It is easy to conclude that the coordinate systems of both markers are in agreement with the image, using Fig. 5.6(b), which allows for a better perception of the position of the elements. When comparing this representation with Fig. 5.5, it is possible to verify that the front axis of the camera is pointing to the markers and the x and y axes correspond to the x and y axes of the image. The x axis starting at the upper left corner of the image being in the horizontal and left direction, the y axis also starts in the upper left corner but has vertical and down direction. This is the same result which is represented in the 3D reconstruction.



(a)



(b)

Figure 5.6: 3D reconstruction of the scenario of dataset 1.

5.1.2 Calibration using the center of the marker

For the same dataset and under the same conditions, the calibration method was tested using only the centers of the markers, as shown with red squares in Fig. 5.7. Since only the corners of the markers are detected by the aruco detection library, the following equation was used to find the center of the markers. This is the same as computing the center of mass considering the cameras all have $m = 1$:

$$x_{\mathrm{CM}} = \frac{m_1 \cdot x_1 + m_2 \cdot x_2 + m_3 \cdot x_3 + m_4 \cdot x_4}{m_1 + m_2 + m_3 + m_4}$$
$$y_{\mathrm{CM}} = \frac{m_1 \cdot y_1 + m_2 \cdot y_2 + m_3 \cdot y_3 + m_4 \cdot y_4}{m_1 + m_2 + m_3 + m_4} \tag{5.1}$$

Figure 5.7 shows the centers of the markers using the initial estimate (blue squares). As verified previously, the reprojection error is very low.

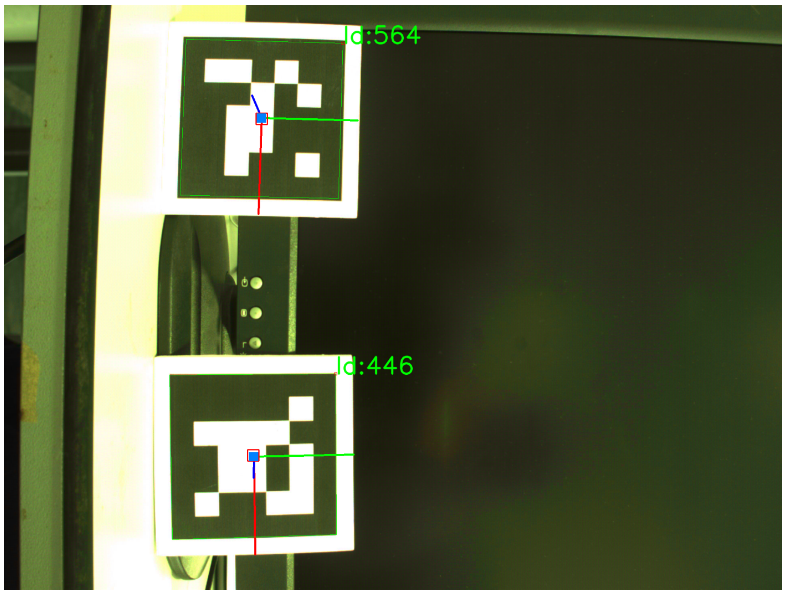


Figure 5.7: Representation of dataset 1 and the markers' center and reprojection of the 3D points on the image using the initial guess.

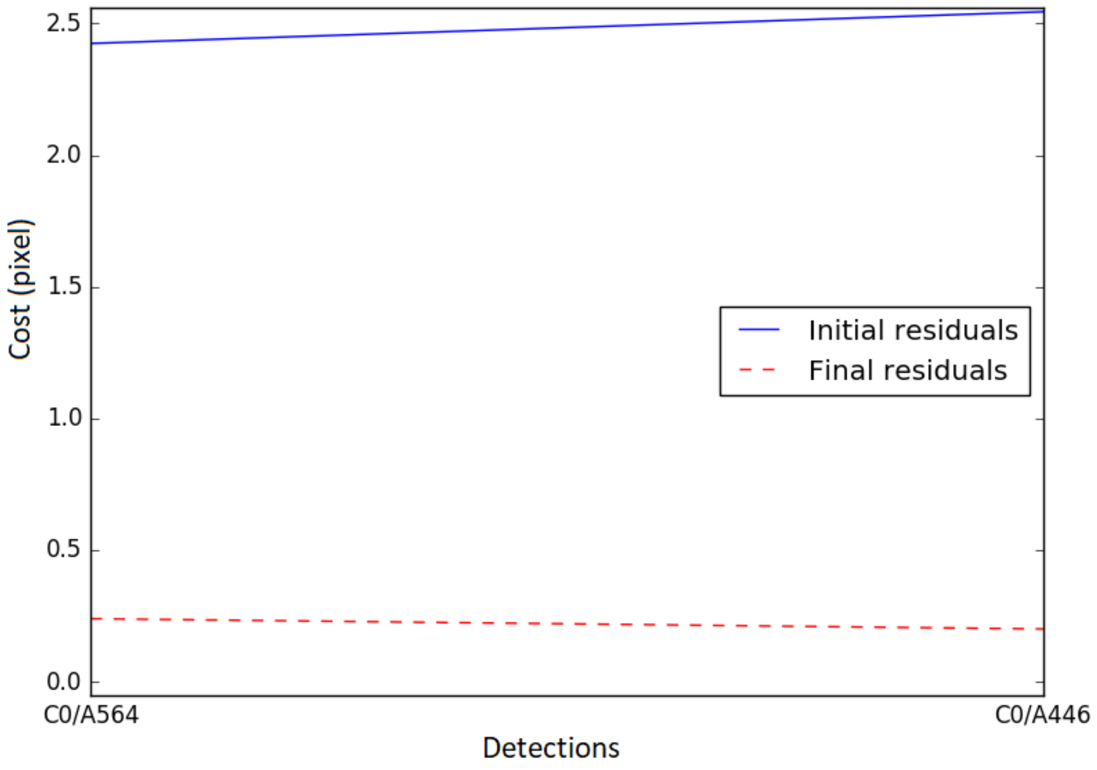


Figure 5.8: Representation of the costs related to each detection.

The reprojection error is minimized using the optimization process. The results of the optimization are represented in Fig. 5.8. Compared to the previous case, there was a higher initial cost of around 2.5 pixels reprojection error. The optimization process in this case is more effective because it reduces the reprojection error to values close to 0.2 pixels.

We have shown that optimization is more successful in the case where only the center of the markers is used. This is associated to the fact that in optimization, where only the center of the markers is used, only the parameters referring to the detection which contains the marker are changed by the optimizer, considering only the error of reprojection of one point. When using the marker's corners, these parameters are modified taking into account a reprojection error of four points. The use of the marker's corners will cause more restrictions on the optimizer, which creates several local minima. If the optimizer starts from an initial estimate with a considerable reprojection error, it may fall to a local minimum, so that it cannot determine a valid solution.

Again, the corresponding points of the center of the markers are projected in the image (represented by yellow circles) using the results obtained from the optimization, as shown in Fig. 5.9. As can be seen, there was a small improvement in relation to the projection that uses the initial estimate (represented by blue square).

Finally, a 3D representation of the dataset was performed, showing in Fig. 5.10. In comparison to the results in Fig. 5.6, it is possible to conclude that the representation is in all similar, and using the arguments used to justify the representation of Fig. 5.6, it is notable that the representation is with an arrangement equivalent to the real arrangement of the markers and camera.

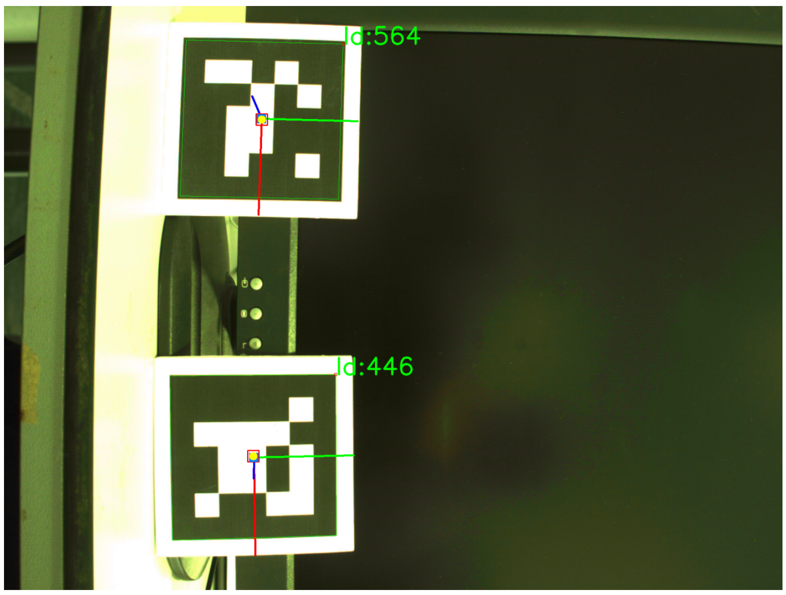
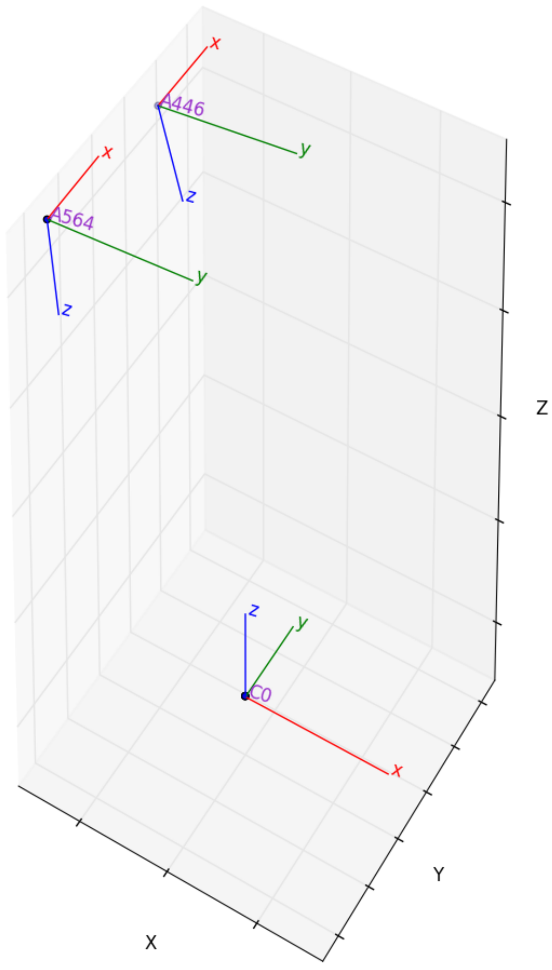
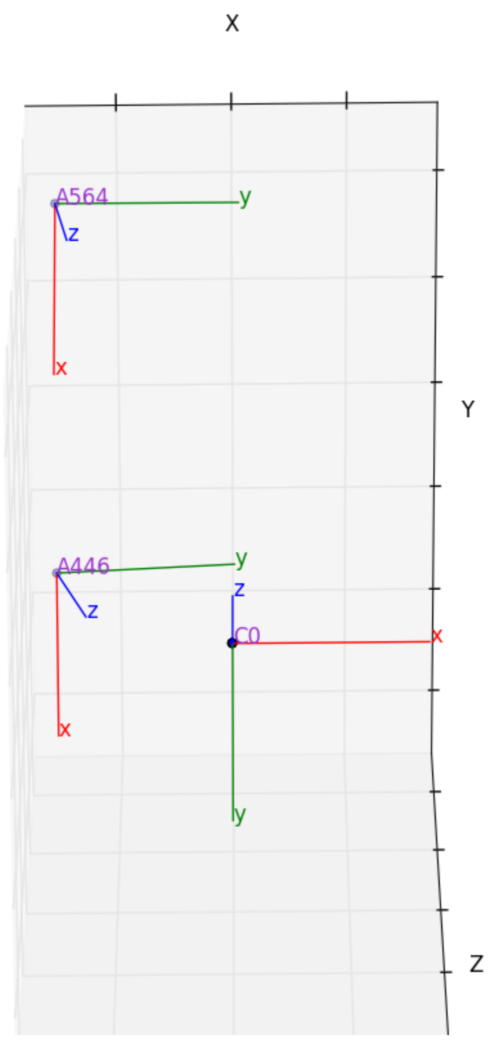


Figure 5.9: Reprojection of the 3D points on the image using the final values resulting from the optimization.



(a)



(b)

Figure 5.10: 3D reconstruction of the scenario of dataset 1.

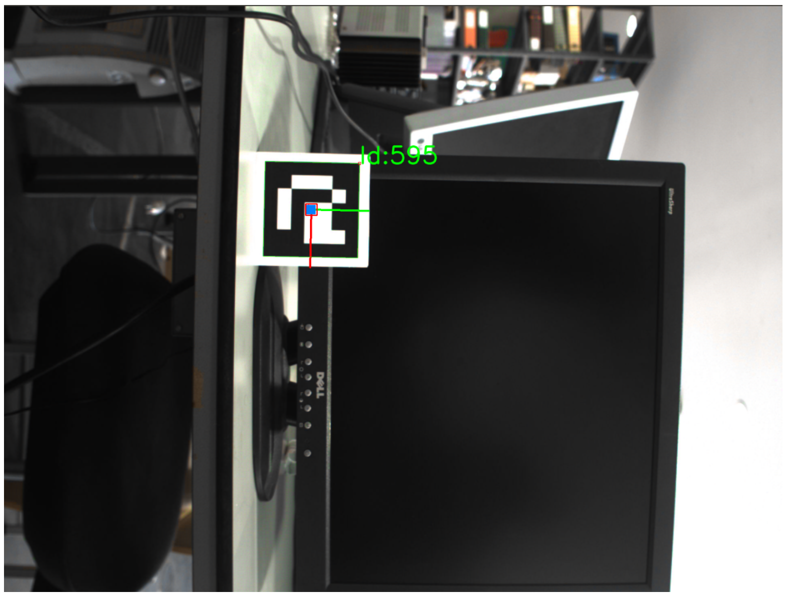
As can be concluded from the tests made in dataset 1, the calibration/optimization process presents better results when only the centers of the markers are used. Thus, for all subsequent datasets only the tests that use the centers of the markers will be presented.

5.2 Dataset 2

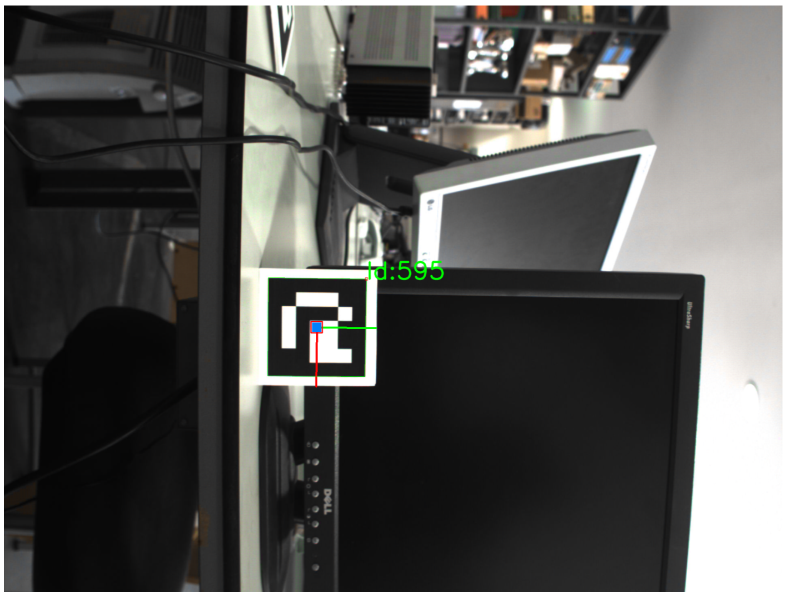
In order to validate the calibration method, an additional simple dataset was developed, consisting of one marker and two cameras, as shown in Fig. 5.11. It is already understood that the problem with this dataset is that of a multi-camera calibration, because there are two cameras with unknown positions in relation to each other. Only one marker will be used to relate the two cameras. This dataset allows us to understand if the pose of the cameras in 3D projection is consistent with the actual position of the cameras.

In Fig. 5.11 the images corresponding to the cameras are shown. One Aruco marker is visible. The centers of the markers are marked as red squares.

The graph of nodes represented in the Fig. 5.12, which was generated according to the detections, makes it possible to verify that the two cameras are connected by a marker. This connection will allow us to estimate where one camera is with respect to the other, relating the transformations obtained on the detections of the markers in the image, which represent the transformation of the marker coordinate system to that of the cameras. Once again, in this dataset, camera 0 was defined as the common reference.



(a) Camera 0



(b) Camera 1

Figure 5.11: Representation of dataset 2 and the center of the markers (red squares) and the reprojection of the 3D points on the image using the initial guess (blue squares).

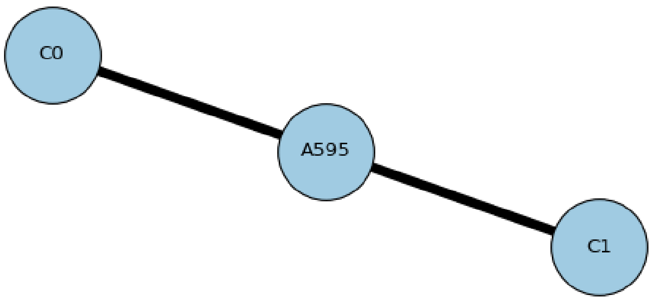


Figure 5.12: Graph node of dataset 2.

Figure 5.11 shows the initial estimate, represented by blue squares. The initial estimate is almost perfect because the blue squares are overlapping the red squares, which represent the center of the Aruco marker.

Although the initial estimate provided good results, the reprojection error minimization process was done. In this dataset there are two detections, C0-A595 and C1-A595. As can be seen in Fig. 5.13, the second detection has a higher cost than the first, with an initial cost of 0.35 pixels, which is considered a low and acceptable cost. Nonetheless, the optimizer was able to reduce this cost to a value close of 0 pixels.

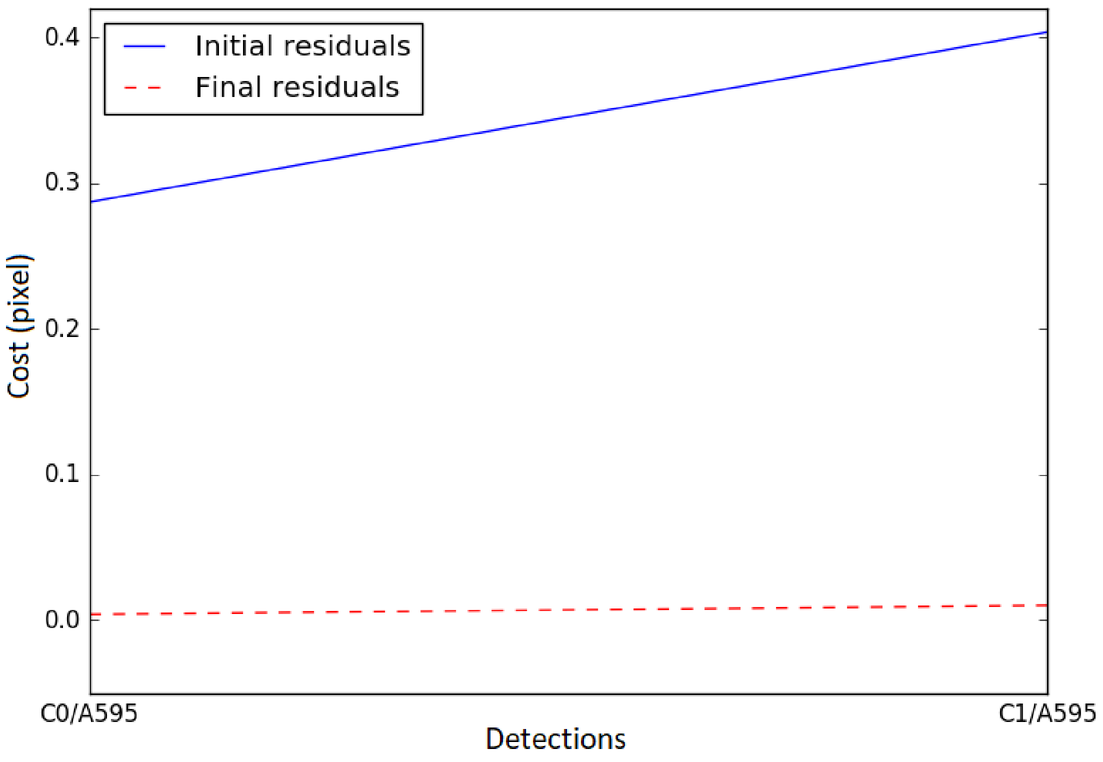
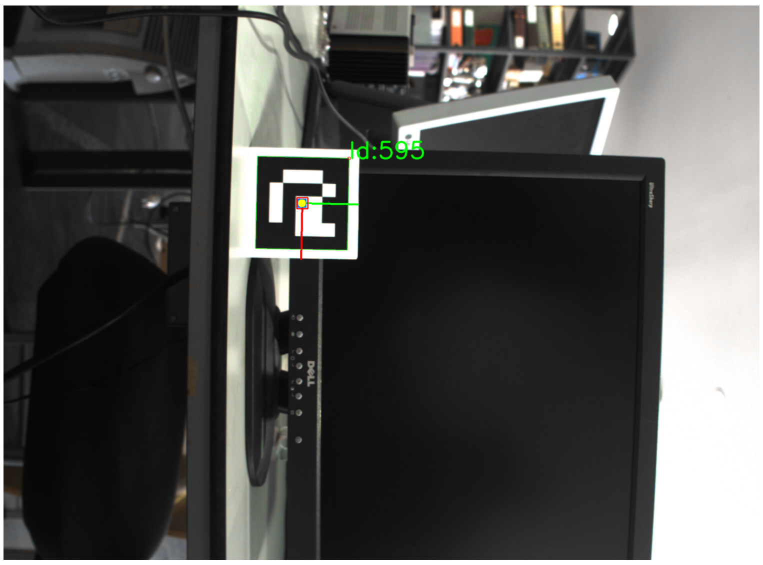
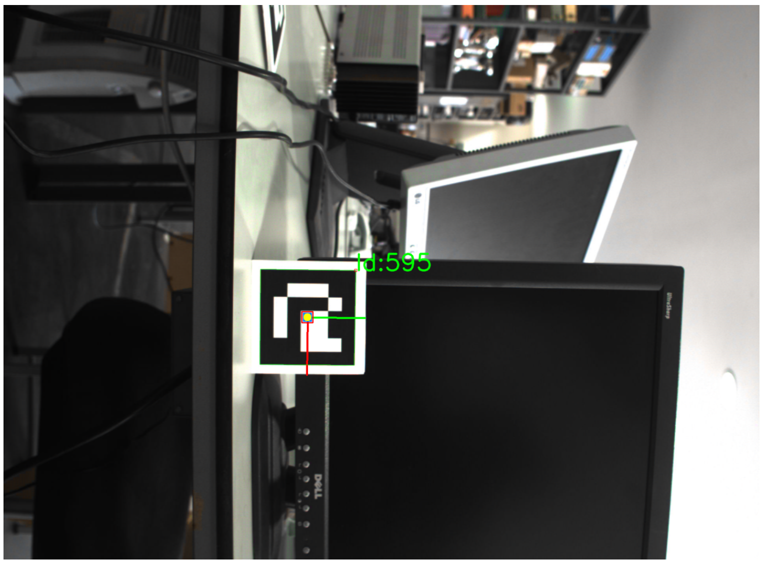


Figure 5.13: Representation of the costs related to each detection.



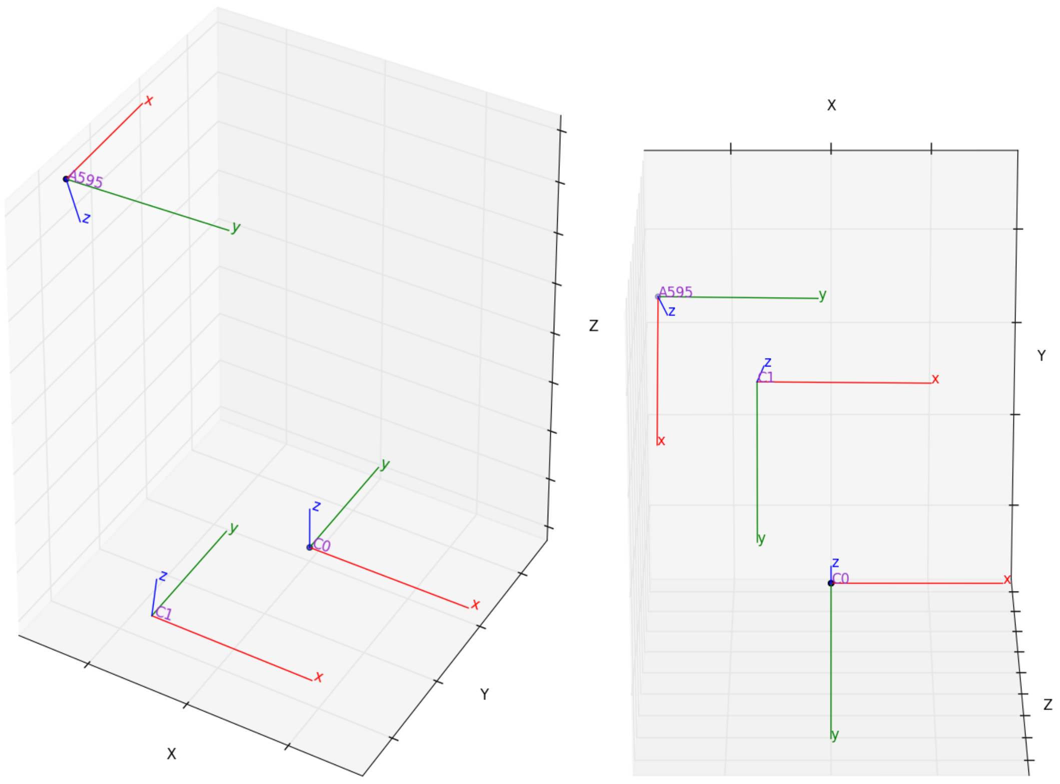
(a) Camera 0



(b) Camera 1

Figure 5.14: Reprojection of the 3D points on the image using the final values resulting from the optimization.

Figure 5.14 represents the points projected on the image using the values optimized (yellow circles). As can be seen, there was not a very significant change in the position of the points because the initial values were already very close to the correct values. Fig. 5.14 also shows that camera 0 is slightly below camera 1 because in the image of camera 0 (Fig. 5.14 (a)) the marker is in a higher position with respect to the image of camera 1 (Fig. 5.14 (b)). In the 3D view of the problem (Fig. 5.15) it is possible to verify that the cameras are positioned in agreement with reality. Fig. 5.15(b) is aligned with the images of Fig. 5.14 which allows a better perception of the 3D arrangement of the cameras, and proves that the disposition of the elements (cameras and marker) is consistent with what was expected.



(a) (b)

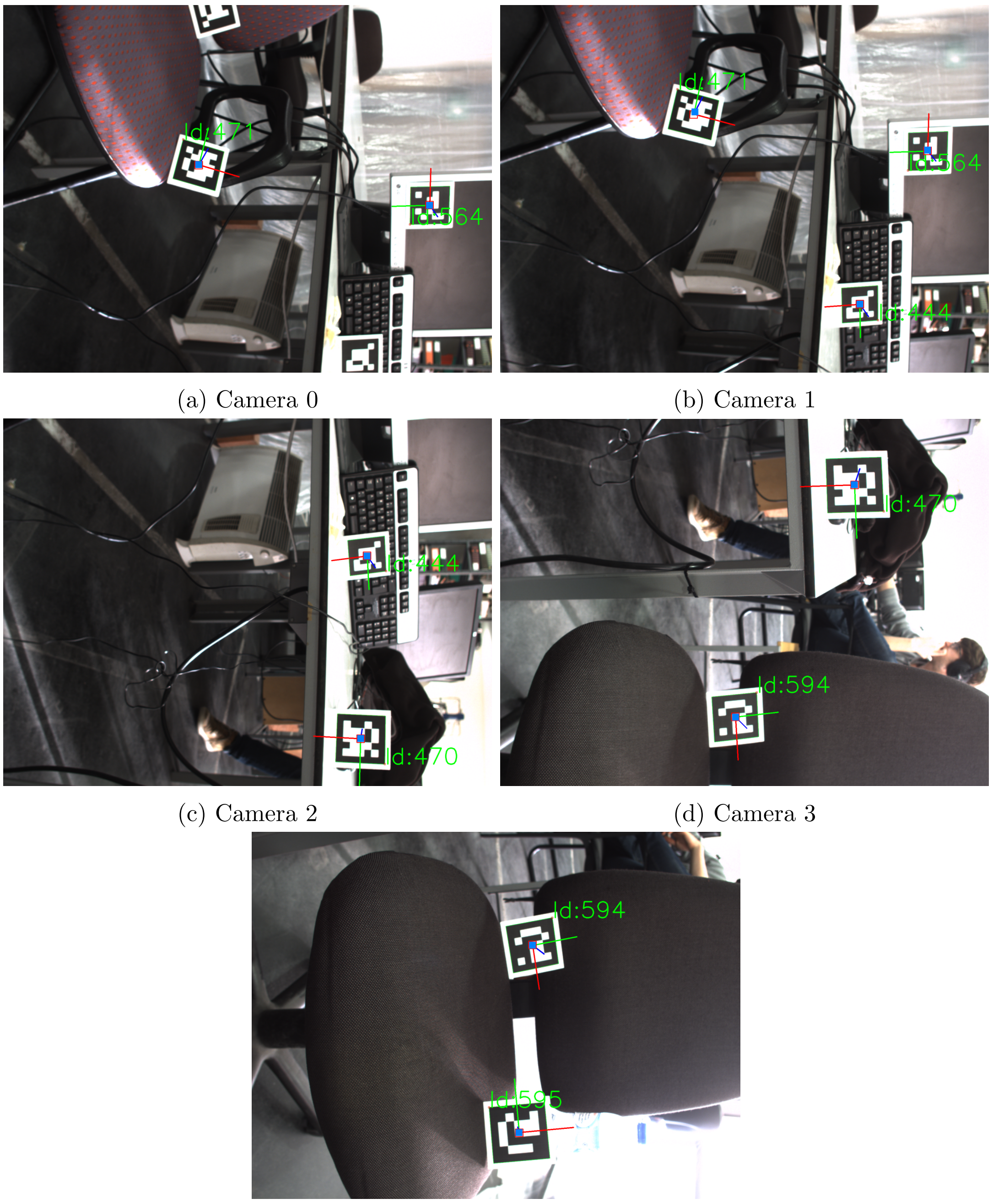
Figure 5.15: 3D reconstruction of the scenario of dataset 2.

5.3 Dataset 3

After validating the calibration method using simple datasets, a more complex dataset containing 5 cameras and 6 markers was created. This dataset is represented in Fig. 5.16. In these images, the markers seen by the cameras are identified, as well as their centers. The detections of the markers by each camera produce a graph represented in Fig. 5.17. In this graph it is possible to verify that cameras C0 and C1 are connected by two markers, and the remaining connections, C1 and C2, C2 and C3, C3 and C4, are connected by only one marker. In this way the whole system is interconnected, so the calibration process is possible. In this dataset, camera 0 was defined as the world reference frame.

Figure 5.16 also illustrates the reprojection of the 3D points on the image using the extrinsic parameters initially estimated. The projection is already very close to the real result, although on detection C1-A471, it is possible to note a small error. However, the other reprojected points are in positions which are very close to the ground truth.

The reprojection errors in some of the detections in this dataset are quite high, so obtaining a good result in the optimization process would be very relevant. The results of the optimization process are represented on graph in Fig. 5.18, and as previously mentioned, the detection C1-A471 presents a large reprojection error of around 12 pixels. Nevertheless, the optimizer was able to obtain good results (represented by the red dotted line). Despite having increased the cost in the first detections, the optimizer reduced the higher costs to much lower values, bringing the costs of all detections to close to 0 pixels. The average initial cost was approximately 2.1 pixels and the optimizer reduced this to 0.35 pixels, which is significant.



(a) Camera 0 (b) Camera 1

(c) Camera 2 (d) Camera 3

(e) Camera 4

Figure 5.16: Representation of dataset 3 and the markers' center (red squares) and reprojection of the 3D points on the image using the initial guess (blue squares).

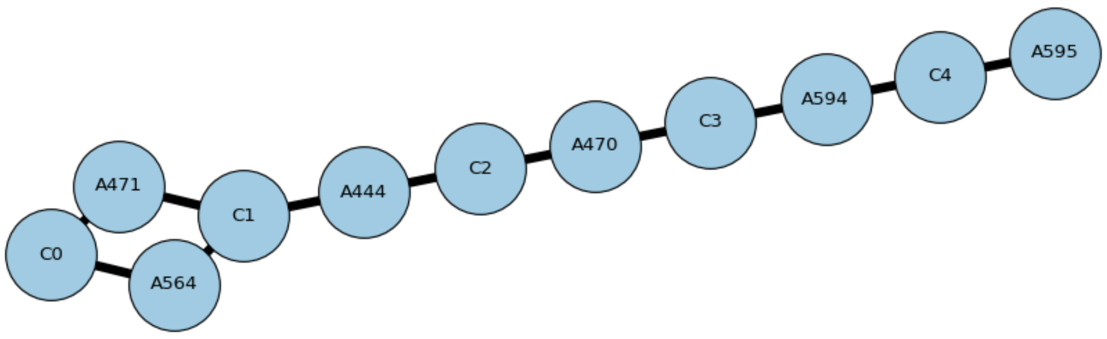


Figure 5.17: Graph node of dataset 3.

Using the values obtained from the optimization, the 3D points were reprojected in the image (represented by the yellow circles), as illustrated in Fig. 5.19. In these results, the improvement of the extrinsic parameters is visible, because all the points reprojected in the image are in close positions with the ground truth detected by image processing.

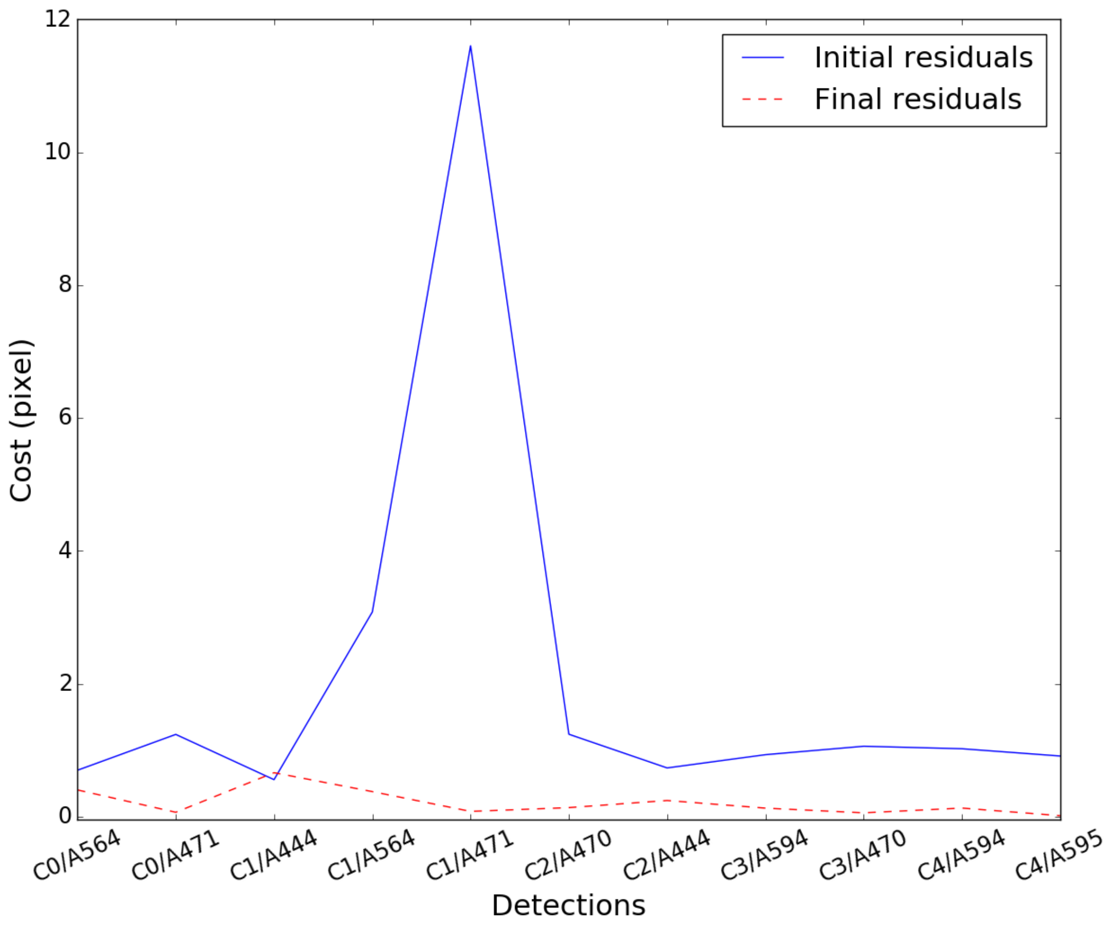
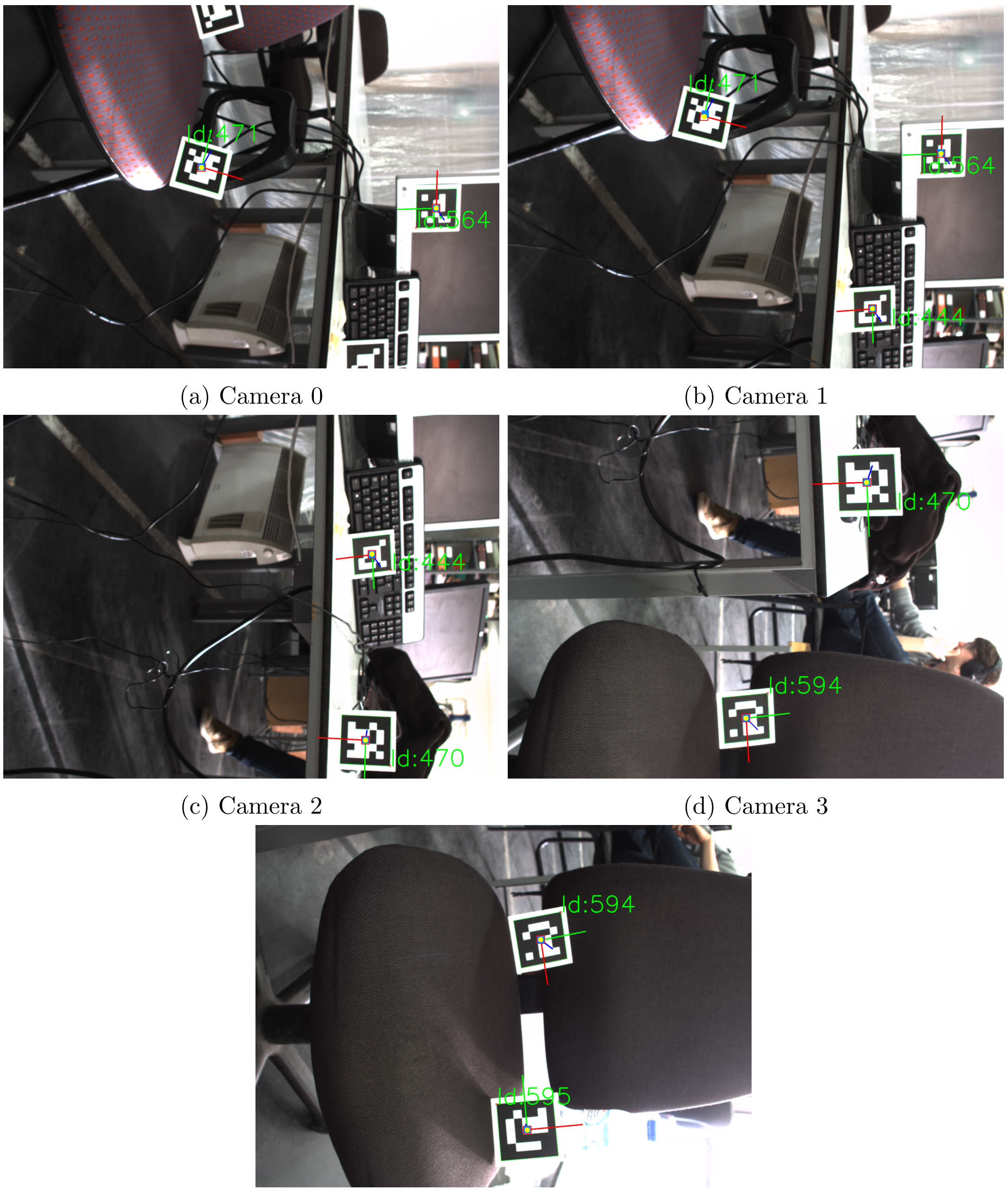


Figure 5.18: Representation of the costs related to each detection.



(a) Camera 0 (b) Camera 1
(c) Camera 2 (d) Camera 3
(e) Camera 4

Figure 5.19: Reprojection of the 3D points on the image using the final values resulting from the optimization.

Figure 5.21 shows a 3D view of the problem, where it is possible to see that the elements of the dataset are in the correct positions according to the real disposition visualized in Fig. 5.20. It should also be noted that all the coordinate systems of the markers in the 3D view are consistent with what was expected. Also, note that the 3D view has a slight rotation on the y-axis. This rotation was purposely done in the 3D view in which the coordinate systems of the cameras were not overlapped on the coordinate systems of the markers.

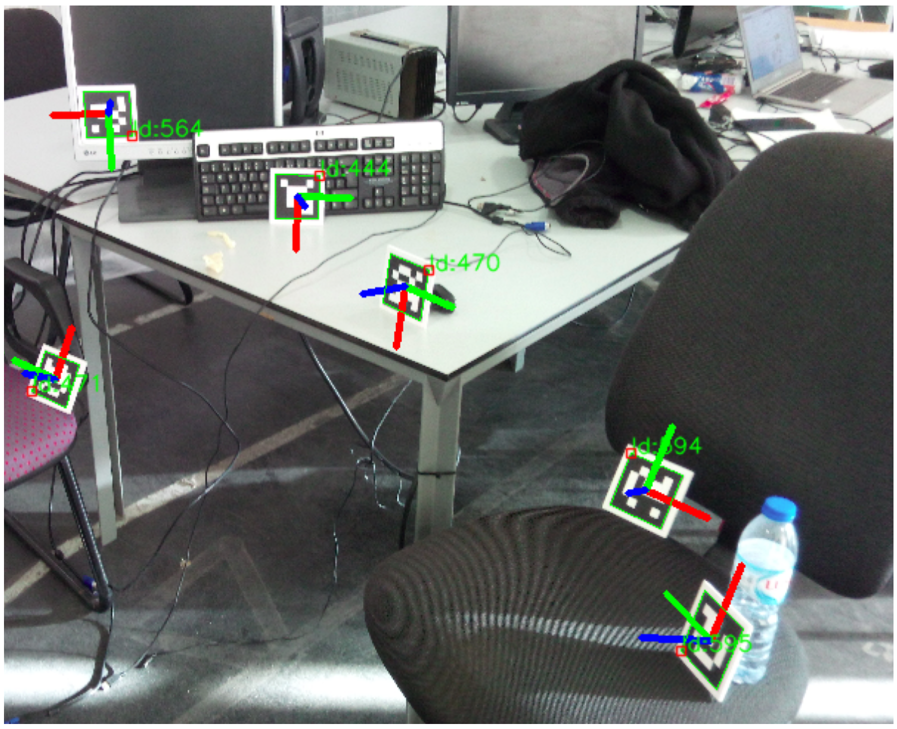


Figure 5.20: Disposition of markers in dataset 3.

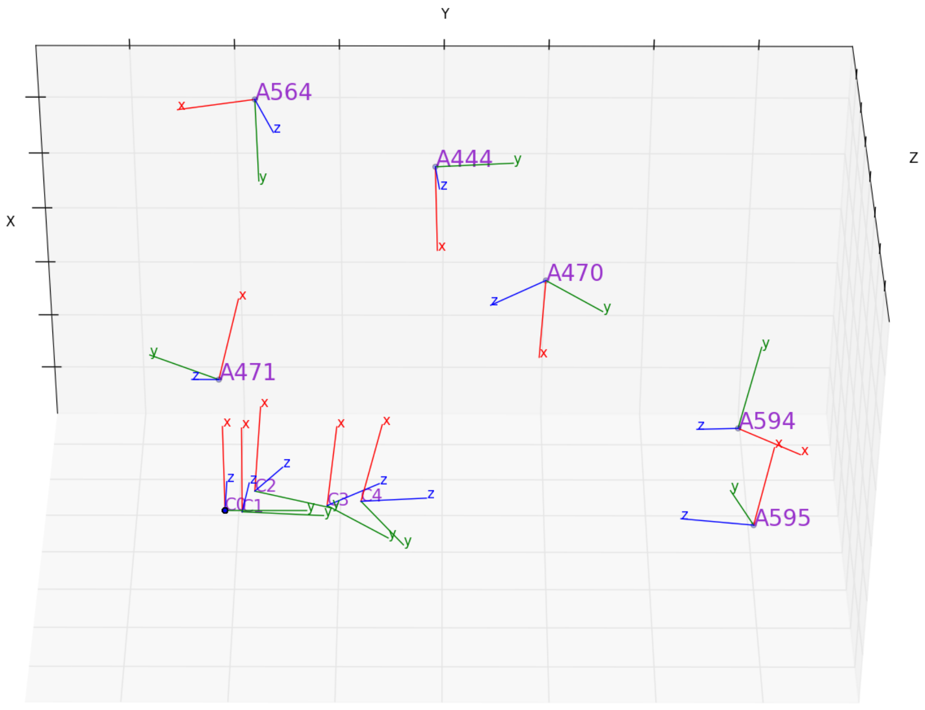


Figure 5.21: 3D reconstruction of the scenario of dataset 3.

5.4 Dataset 4

In the previous datasets, analysing the reprojection of the points in the image and showing a 3D view of the elements present in each dataset allowed the verification of the validity of the implemented calibration method. However, this validation is only an empirical, qualitative validation.

For this reason, it was necessary to create a dataset in which the poses of the elements of the dataset were known. This information is difficult to obtain for cameras. However, it is possible to create a dataset in which the markers are in known positions.

For this, a marker board was created, as shown in Fig. 5.22, where it is known that the markers are all in the same plane. Knowing the lateral size of each marker is 100 mm and that these are spaced at 20 mm, both in vertical and horizontal directions, it is possible then to have the arrangement of the markers with a ground truth value for the positions of all the markers.

To obtain the images of this dataset the ASUS ZenFone AR was used. This equipment allows the making of 3D reconstructions of the surrounding space, and as previously mentioned in section 3.2.2, it acquires several images from several perspectives. In fact, for this dataset two tests were done, both tests use the same marker board, however, the first dataset contains 104 images and the second test contains 38 images. All the analysis of the dataset will be carried out for the test with 104 cameras and in the end the values of the average errors of the 3D reprojection of both will be compared.

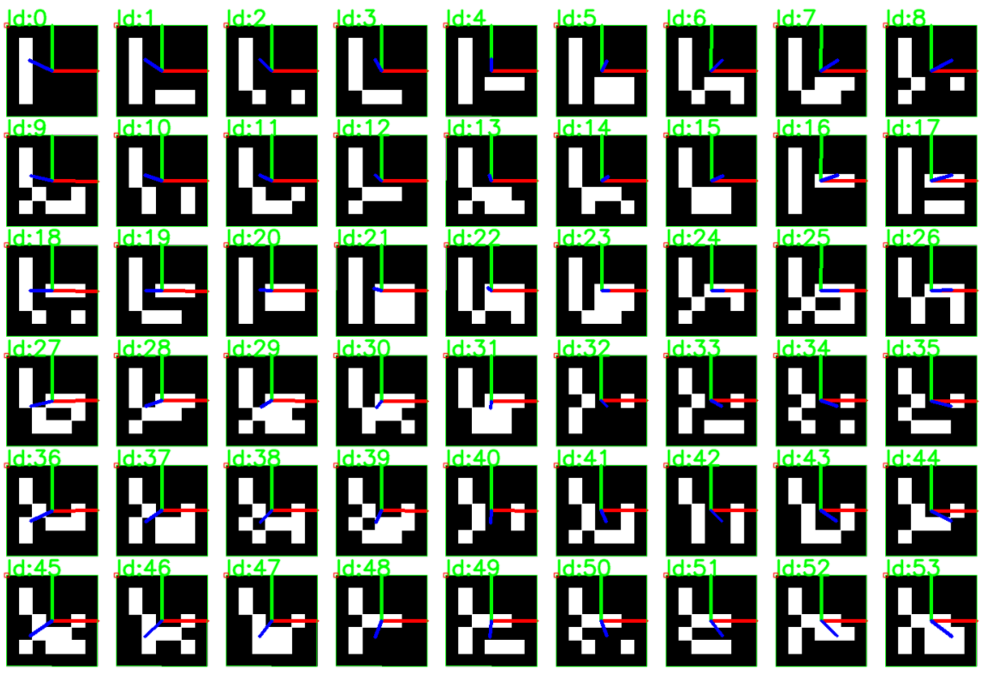


Figure 5.22: Marker board with real dimensions.

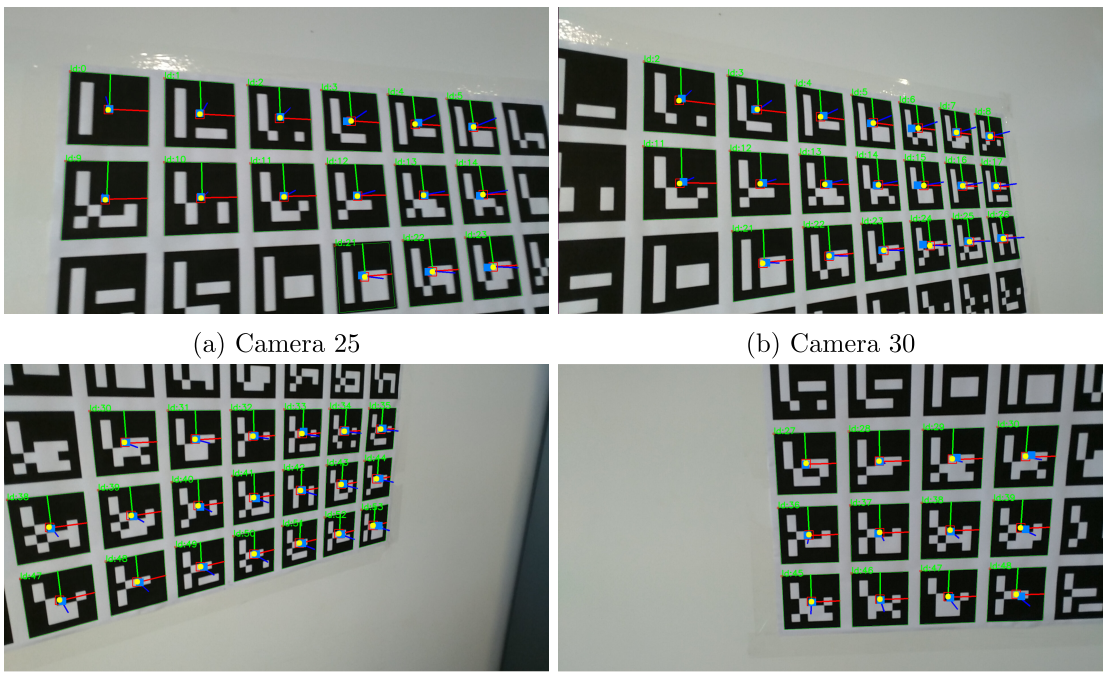
In order to facilitate the analysis of the dataset, marker "A0" was defined as the global reference frame, which allows to find the positions of all the others. Thus, the center of all markers must be located in the xy plane with z=0, and the locations x and y can be easily calculated by knowing the dimensions of the marker board, where the centers are spaced 120 mm in the x and y directions.

The generated graph for this dataset is too complex since it consists of 104 nodes referring to the cameras, plus 54 nodes referring to the markers. Since each camera visualises about 20 markers, the number of existing links is high. Due to this, the representation of the graph of nodes is practically imperceptible when represented in a small space, so it is not shown in this document.

Figure 5.23 represents some of the images present in this dataset. It is possible to see the reprojected points which use the initial estimate (represented by the blue squares) and those that use the values obtained after the optimization (represented by the yellow circles). It is also possible to verify that the initial estimate has a large reprojection error with respect to the ground truth. However, the optimizer achieved good results, managing to overlay the reprojected points with the ground truth.

The cost for each detection graph is shown in Fig. 5.24. There was a significant cost reduction over all the detections. The average cost of the initial estimate was 9.6 pixels, and the optimizer was able to reduce it to a cost of 0.85 pixels.

The 3D representation of the problem is shown in Fig. 5.25. To have the perception if the cameras are located in the correct position is almost impossible. However, it is quite perceptible that the representation of the center of the markers is consistent with reality. Hereupon, it is necessary to analyse the numerical results of this test to prove that the calibration method is reliable.



(a) Camera 25 (b) Camera 30

(c) Camera 37 (d) Camera 46

Figure 5.23: Representation of some images. Reprojection of the points using the initial estimate (blue squares). Reprojection of the points after the optimization (yellow circles).

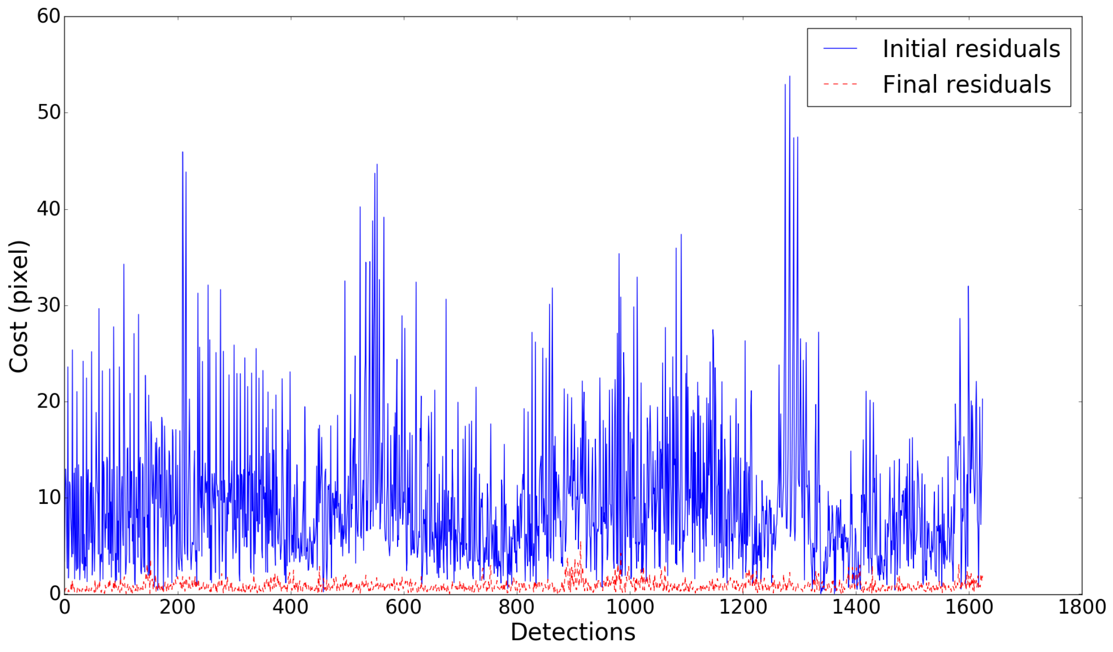


Figure 5.24: Representation of the costs related to each detection.

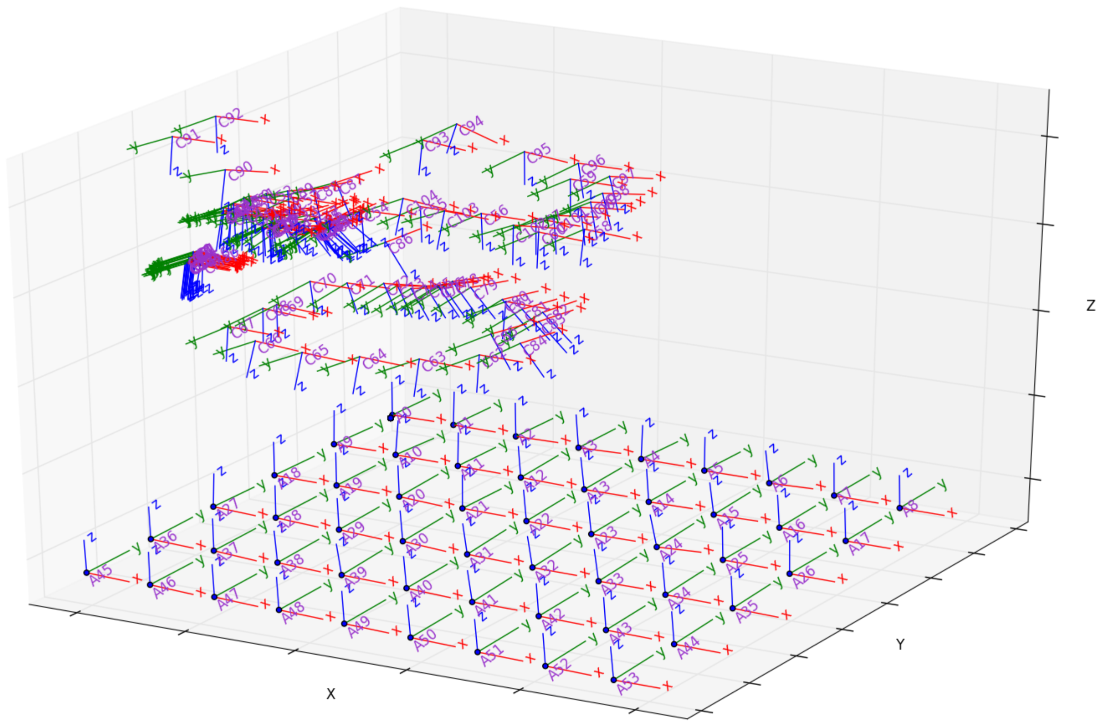


Figure 5.25: 3D reconstruction of scenario of the dataset 4.

As already mentioned, marker "A0" is the global reference frame, that is, its center will be the origin of the global reference frame. Thus, the real position of all markers in relation to the global reference is already known. The average error of the 3D position of all centers of the markers is computed before and after the optimization process. So, it is necessary to compute the position of the points using the first guess and using the result of optimization.

Having these three sets of points: the real position of the points, the position of points using the initial estimate and the position of the points using the data obtained from the optimization, it is possible to calculate the average error before and after the optimization using Eq. (5.2). Equation 5.2 computes the sum of the Euclidean distances of the real center of the marker and the center computed, where N_m is the number of markers and divides the sum of the Euclidean distances by the total number of markers, giving the average error (AE).

$$\mathrm{AE} = \frac{\sum_{i=1}^{N_m} \sqrt{(x^i_{\text{computed}} - x^i_{\text{real}})^2 + (y^i_{\text{computed}} - y^i_{\text{real}})^2 + (z^i_{\text{computed}} - z^i_{\text{real}})^2}}{N_m} \tag{5.2}$$

Table 5.1 shows the average error for the initial estimate and for the optimization result, where the noise over the initial estimate is also varied.

		% of noise				
Test with:	Parameters	Nd	1	5	10	15
104 cameras	**Initial estimate**	0.0243	0.0247	0.0317	0.0457	0.0557
	Optimization result	0.0283	0.0250	0.0307	0.0314	0.0416
38 cameras	**Initial estimate**	0.0120	0.0123	0.0199	0.0393	0.0520
	Optimization result	0.0117	0.0097	0.0100	0.0164	0.0302

Table 5.1: Average Error (meters).

From the results obtained, it can be concluded that the optimization process can reduce the average error of the 3D projection, even if in the initial estimate there is some noise. This reduction is bigger when the noise present in the initial estimate is higher. However, for the test which uses 104 images, the optimization process practically maintained the average error for the cases where no noise was applied and for the case where 1% of noise was applied. Even so the values obtained are quite appreciable.

Chapter 6

Conclusions and future work

6.1 Conclusions

Today, multisensor platforms have many applications. However, it is necessary to calibrate these sensors to obtain a good performance from the platform. There is currently no fully reliable multi-sensor calibration method yet, and almost all of them require manual intervention from the user. Hence the need to develop a calibration method that allows us to do it anywhere, where it is only necessary to spread Aruco markers in the surrounding space for the process to work. Therefore, the method implemented in this work is of high relevance, because it is a method that does not require any manual intervention from the user to function, it can calibrate a large number of camera sensors using only some Aruco markers and it produces good results.

It is important to mention, before drawing any conclusion, that no datasets were assembled in the ATLASCAR since only one camera has been installed in the vehicle for now. The reason for not having placed any datasets in the ATLASCAR is not only the previously mentioned one, but also the fact that this calibration method can be applied to any type of platform with any number of cameras. Due to its versatility, this method can be tested here in a large number of systems.

This work allows us to validate a method of calibration of N cameras at the same time, without needing any intervention from the user, using the Aruco markers. This calibration process is fully automated. It is only necessary to ensure that all cameras have at least one marker that is also visible by another camera, so that it is possible to establish connections between them, achieving a fully connected graph.

With this work, it was also possible to show the accuracy of the values obtained in the detection of the Aruco markers, which allows us to obtain excellent first guesses, compared to the information acquired by the *aruco* module of OpenCV.

Sometimes the first guess did not present very precise values and to solve this problem an optimization method based on the Bundle Adjustment was used. This method uses a cost function based on the Euclidean distance between the image points and the 3D points reprojected in the image and presented very reliable results. In the chapter of the results of the several calibration tests, it was possible to verify very significant reductions in the reprojection error, which validates the optimization method used.

The 3D projections for each dataset represented above are consistent with the actual arrangement of the elements, which also validate the calibration method presented.

When the calibration method was subjected to a test where the real positions of the markers were known, to enable later comparison with the results of the calibration, this was always successful, presenting a good first guess. It was also possible to verify that even if the first guess showed some noise, the optimizer would be able to find a better solution.

6.2 Future Work

In the future, this work could be continued, because it proved to have good results, and above all, proved to be a versatile calibration method, with numerous applications.

One of the areas to be developed using this method would be to explore the bounds that can be imposed on the parameter vector that is optimized. The setting of bounds in the vector is a very powerful tool because with it, it is possible to address the redundancies of the problem, and thus, to make the problem converge to the correct solution more easily.

Another area would be the application of this calibration method in a dataset taken from the ATLASCAR. With this it would be possible to compare this calibration method with the calibration methods already implemented, and thus, to draw some conclusions. This test would also contemplate the fact that the calibration results can be used and tested on a platform with real use.

This method only calibrates cameras, so other work could be based on the expansion of this method, where in addition to the cameras, other types of sensors like LRFs would also be calibrated. In this way the calibration method would be more global.

The inclusion of the LRF sensors in the method implemented will promote some changes. Initially, the 3D LRFs have to detect the Aruco markers, e.g. finding features with square form in the LRF data. Next, the enhancement of the center of the Aruco markers becomes possible doing a treatment of the point cloud. However, even detecting the centers of the markers, you should need to know their identifier, which is not possible. Therefore, another solution would be to carry out a comparison between both the point clouds generated by the cameras as well as by the LRFs and find similarities assigning an identifier to the markers.

Nevertheless, the earlier method has not much utility for 2D LRFs. For this, a RADLOCC based method based on [40] can be implemented. This method uses a chessboard as a calibration target to do a camera to 2D LRF calibration, where the chessboard is easily detected by the cameras as well as by the 2D LRFs. The detection of the chessboard by the 2D LRF is based on the algorithm implemented in [25]. Finally, the pose of the two sensors between each other is possible to obtain.

References

[1] *Generate ArUco Markers for printing.* https://tn1ck.github.io/aruco-print/.

[2] *Projeto ATLAS - Universidade de Aveiro*, 2003. https://www.http://atlas.web.ua.pt/.

[3] *Flea3 FL3-GE-28S4 Camera*, 2012. https://www.visiononline.org/vision-resources-details.cfm/vision-resources/Flea3-FL3-GE-28S4-Camera/content_id/3694.

[4] *Bundle adjustment*, 2018. https://en.wikipedia.org/wiki/Bundle_adjustment.

[5] *Quaternions and spatial rotation*, 2018. https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation.

[6] *Slerp*, 2018. https://en.wikipedia.org/wiki/Slerp.

[7] Sameer Agarwal, Noah Snavely, Steven Seitz, and Richard Szeliski. Bundle Adjustment in the Large Computer Vision. *Eccv*, 6312:29–42, 2010.

[8] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 102–115, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[9] Miguel Almeida. Reconstrução 3D e calibração de lasers no AtlasCar. 2011.

[10] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, Dec 2008.

[11] J. Y. Bouguet. *Camera Calibration Toolbox for Matlab*, 2010. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[12] Gary Bradski. *The OpenCV Library*, 2000. http://www.drdobbs.com/open-source/the-opencv-library/184404319.

[13] Andrew Brooks. *Graph Optimization with NetworkX in Python*, 2017. https://www.datacamp.com/community/tutorials/networkx-python-graph-tutorial.

[14] Daniel Coimbra e Silva. LIDAR Target Detection and Segmentation in Road Environment. page 104, 2013.

[15] OpenCV documentation. *Camera Calibration and 3D Reconstruction*, 2018. `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.

[16] R. Duda and P. Hart. Pattern Classification and Scene Analysis. *John Wiley and Sons Inc*, 1973.

[17] E. Fernández-Moral, V. Arévalo, and J. González-Jiménez. Extrinsic calibration of a set of 2D laser rangefinders. 2015(C):2098–2104, 2015.

[18] Martin a Fischler and Robert C Bolles. Paradigm for Model. *Communications of the ACM*, 24(6):381–395, 1981.

[19] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* 2003.

[20] S Garrido-Jurado, R Muñoz-Salinas, F J Madrid-Cuevas, and M J Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[21] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, 17:883–893, 1980.

[22] Carlos Guindel, Jorge Beltrán, David Martín, and Fernando García. Automatic Extrinsic Calibration for Lidar-Stereo Vehicle Sensor Setups. 2017.

[23] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[24] Lionel Heng, Gim Hee, and Lee Marc. Self-Calibration and Visual SLAM with a Multi-Camera System on a Micro Aerial Vehicle. (2009):1–19, 2013.

[25] Abdallah Kassir and Thierry Peynot. Reliable automatic camera-laser calibration. *Proceedings of the 2010 {Australasian} {Conference} on {Robotics} & {Automation}*, 2010.

[26] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*, pages 2969–2976, June 2011.

[27] Kurt Konolige. Kurt konolige : Sparse sparse bundle adjustment 1 sparse sparse bundle adjustment. 2010.

[28] Kiho Kwak, Daniel F. Huber, Hernan Badino, and Takeo Kanade. Extrinsic calibration of a single line scanning lidar and a camera. *IEEE International Conference on Intelligent Robots and Systems*, pages 3283–3289, 2011.

[29] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[30] Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers, 06 2013.

[31] Manolis I. A. Lourakis and Antonis A. Argyros. Sba. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

[32] Donald W Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[33] McKinsey&Company. *Self-driving car technology*, 2018. `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/self-driving-car-technology-when-will-the-robots-hit-the-road`.

[34] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950, April 2007.

[35] OpenCV. *Detection of ArUco Markers*, 2015. `https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html`.

[36] Gaurav Pandey, James R Mcbride, Silvio Savarese, and Ryan M Eustice. Automatic Targetless Extrinsic Calibration of a 3D Lidar and Camera by Maximizing Mutual Information. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2053–2059, 2012.

[37] Marcelo Pereira, Vitor Santos, and Paulo Dias. Automatic Calibration of Multiple LIDAR Sensors Using a Moving Sphere as Target. In Luís Paulo Reis, António Paulo Moreira, Pedro U Lima, Luis Montano, and Victor Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 477–489, Cham, 2016. Springer International Publishing.

[38] Marcelo Pereira, David Silva, Vítor Santos, and Paulo Dias. Self calibration of multiple LIDARs and cameras on autonomous vehicles. *Robotics and Autonomous Systems*, 83:326–337, 2016.

[39] Marcelo Silva Pereira. Automated calibration of multiple LIDARs and cameras using a moving sphere. 2015.

[40] Qilong Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2301–2306.

[41] Jan Quenzel, Nils Papenberg, and Sven Behnke. Robust extrinsic calibration of multiple stationary laser range finders. *IEEE International Conference on Automation Science and Engineering*, 2016-Novem(August):1332–1339, 2016.

[42] Technical Reference. Flea3. 2017.

[43] Kumar Robotics. *flea3*, 2017. `https://github.com/KumarRobotics/flea3`.

[44] ROS. *About ROS*, 2018. `http://www.ros.org/about-ros/`.

[45] Jorg Rowekamper, Michael Ruhnke, Bastian Steder, Wolfram Burgard, and Gian Diego Tipaldi. Automatic extrinsic calibration of multiple laser range sensors with little overlap. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015-June(June):2072–2077, 2015.

[46] Radu Bogdan Rusu and S Cousins. 3D is here: point cloud library. *IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[47] V. Santos, J. Almeida, E. Ávila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, and P. Stein. ATLASCAR - Technologies for a computer assisted driving system on board a common automobile. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2008:1421–1427, 2010.

[48] Vítor Manuel Ferreira Santos. Sebenta Robótica Industrial 2003-2004. page 166, 2004.

[49] TechRepublic. *Autonomous driving levels 0 to 5: Understanding the differences*, 2016. `https://www.techrepublic.com/article/autonomous-driving-levels-0-to-5-understanding-the-differences/`.

[50] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment A Modern Synthesis. *Vision Algorithms: Theory and Practice*, 1883:298–372, 2000.

[51] David Tiago Vieira da Silva. Multisensor Calibration and Data Fusion Using LIDAR and Vision. page 107, 2016.

[52] Ying-mei Wei, Lai Kang, Bing Yang, and Ling-da Wu. Applications of structure from motion: a survey. *Journal of Zhejiang University SCIENCE C*, 14(7):486–494, jul 2013.

[53] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3930–3935, April 2005.