Universidade de Aveiro Departamento de Engenharia Mecânica
2015

Marcelo Silva Pereira

Calibração automática de múltiplos LIDARs e câmaras usando uma esfera em movimento
Automated calibration of multiple LIDARs and cameras using a moving sphere

Universidade de Aveiro Departamento de Engenharia Mecânica
2015

Marcelo Silva Pereira

Calibração automática de múltiplos LIDARs e câmaras usando uma esfera em movimento
Automated calibration of multiple LIDARs and cameras using a moving sphere

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 Vítor Manuel Ferreira dos Santos, Professor associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Paulo Miguel de Jesus Dias, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri / The jury

Presidente / President	**Prof. Doutor Jorge Augusto Fernandes Ferreira** Professor Auxiliar da Universidade de Aveiro
Vogais / Committee	**Doutor Cristiano Premebida** Investigador da Universidade de Coimbra - Faculdade de Ciências e Tecnologia (arguente)
	Prof. Doutor Vítor Manuel Ferreira dos Santos Professor associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Em primeiro ligar um especial agradecimento ao professor Vítor Santos pela sua entrega ao projeto como orientador. Ao longo deste percurso sempre demonstrou uma dedicação e um acompanhamento exemplar. Denota-se que tem uma paixão enorme pelo que faz, sendo sempre uma fonte de boa disposição e constante entusiasmo que é contagiante, provindo uma motivação extra para atravessar barreiras ao longo do caminho.

Não menos importante, outro especial agradecimento ao professor Paulo Dias. A sua dedicação, exigência e o vasto conhecimento no que diz respeito a este trabalho foram de extrema importância. As numerosas discussões que tivemos durante a realização deste trabalho foram sempre enriquecedoras, ajudando a ultrapassar os obstáculos à realização deste trabalho.

À minha família, quero agradecer o apoio incondicional e a oportunidade que me deram de poder realizar o curso de Engenharia Mecânica. Sem eles nunca teria sido possível.

Por fim, queria agradecer aos meus colegas do LAR por todos os momentos de boa disposição passados. O bom ambiente entre esta pequena família também se torna essencial à realização de um bom trabalho.

Palavras-chave

Nuvem de pontos; reprojeção 3D; fitting de dados 3D; transformação de corpo rígido.

Resumo

Veículos autónomos têm atraído muito interesse nos últimos anos devido ao seu potencial impacto na sociedade, o que tem impulsionado esta área para estudos e desenvolvimentos constantes. Uma vez que os sistemas de perceção são extremamente importantes na navegação autónoma, a sua complexidade leva a um incremento do número de sensores a bordo (composto normalmente por sensores LIDAR, câmaras entre outros) juntamente com o aumento da sua diversidade, o que aumenta a preocupação sobre a calibração de sensores. Os métodos de calibração são normalmente manuais ou semi-automáticos e requerem intervenção de um utilizador. Poucos métodos automáticos estão disponíveis, e mesmo os que existem são normalmente baseados em processos complexos e dispositivos dispendiosos. Este trabalho apresenta um novo método de calibração automático usando uma bola como alvo para extrair correspondências entre sensores. O processo de calibração consiste em mover a bola permitindo a deteção do seu centro ao longo de sucessivas posições por todos os sensores a serem calibrados. Este estudo envolve a calibração de sensores LIDAR 2D e 3D, e câmaras. A segmentação em 2D usa um algoritmo baseado nas propriedades geométricas de um arco. Em 3D, a Point Cloud Library (PCL) sample consensus module é usado para identificar e localizar a bola. Finalmente, OpenCV é usado para calibrar o sistema stereo e computar a imagem de disparidade e a sua re-projeção 3D, resultando numa nuvem de pontos 3D. Durante o movimento da bola, é criada uma nuvem de pontos dos centros da bola para cada sensor. Finalmente, cada nuvem de pontos é alinhada com um sensor de referência. O resultado final do processo é a transformação de corpo rígido de cada sensor com respeito ao sensor de referência. O método foi testado quer em laboratório quer com um veículo em tamanho real (AtlasCar). As relativas calibrações entre sensores assegura muito bons resultados que são avaliados pela consistência da performance da deteção por todos os sensores calibrados. Outra característica adicional nesta solução é a sua flexibilidade ao permitir a calibração de diferentes LIDARs e câmaras.

Keywords Point cloud; 3D re-projection; 3D data fitting; rigid body transformation.

Abstract Autonomous vehicles have attracted great interest in the past years due to their potential impact on society, which has been pushing this area into continuously study and development. Since the perception systems are extremely important in autonomous navigation, their complexity leads to an increment of the number of sensors on board (composed commonly by LIDAR, cameras and other sensors) along with the increase of their diversity, which raised concerns about sensor calibration. Calibration methods are usually manual or semi-automatic and require user intervention. Few automatic methods are available, and even the existent methods are normally based in complex processes and expensive devices. This work presents a new automatic calibration method using a ball as target to extract correspondences between sensors. The process of calibration consists of moving the ball allowing the detection of its center along successive positions by all the sensors to be calibrated. This study involves the calibration of 2D and 3D LIDAR sensors, and cameras. Segmentation in 2D uses an algorithm based on the geometric properties of an arc. In 3D, the Point Cloud Library (PCL) sample consensus module is used to identify and locate the ball. Finally, OpenCV is used to calibrate a stereo system and compute the disparity image and its 3D re-projection, resulting in a 3D point cloud. During ball motion, a point cloud of the ball centers is created for each sensor. Finally, all the point clouds are aligned with a reference sensor. The final result of the process is the rigid body transformation of each sensor with respect to the reference frame. The method was tested both in laboratory experiments and in a real full size vehicle (AtlasCar). The relative calibration among all sensors yields very good results that are evaluated by the consistency of the detection performed by the calibrated sensors. Another additional feature of this solution is its flexibility by permitting the calibration of several different LIDARs and cameras.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many vehicles with autonomous navigation capabilities, and also many advanced driver assistance systems (ADAS), rely on LIDAR, VISION and RADAR based devices. In particular VISION and LIDAR since they complement each other, resulting frequently in accuracy and efficiency. Moreover, most of the developed systems use multiple sensors simultaneously, sometimes even combining different types of LIDAR sensors (1D, 2D, n×2D, 3D). Thus, when there is more than one sensor in the same setup, an accurate extrinsic calibration is increasingly important so the data from the different sensors can be combined or fused in a common frame.

The majority of the existing methods are manual or semi-automatic. Manual approaches have large errors associated to it that may affect the efficiency of the perception system, in special when the sensors have a long working range, where even small errors can result in significant distortions. For the remaining approaches, the most common approach is establishing correspondences in the data from the devices. Those correspondences are normally introduced through manual inputs or using a target object, which in general has a well know geometry.

Autonomous vehicles have attracted great interest in the past years due to their potential impact on society, which has been pushing this area into continuously study and development. Since the perception systems are extremely important in autonomous navigation, their complexity leads to an increment of the number of sensors on board (composed commonly by LIDAR, cameras and other sensors) along with the increase of their diversity. Thus, the typical calibration methods are not enough, it is necessary a more generalized and accurate methods.

To solve the necessity of a calibration method, a fully automatic method is developed in this work, having no requirement of manual measurements or manual correspondences in the data. Instead of those common approaches, a ball is used as a calibration target allowing the detection of its center by the different sensors.

1.1 The ATLAS Project

ATLAS [1] is a project created by the Group of Automation and Robotics at the Department of Mechanical Engineering of the University of Aveiro, Portugal. This project started in 2003 with the purpose of competing in autonomous driving competitions taking place at the Portuguese National Robotics Festival. Since then, several robots of

the ATLAS series were developed (Figure 1.1) and had a continuous success in those competitions and won many prizes.

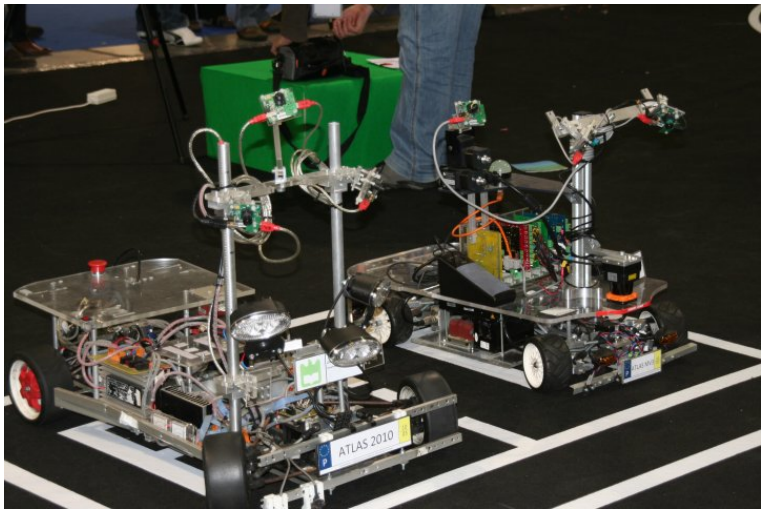


Figure 1.1: ATLAS robots used in the autonomous driving competitions.

With all the vast experience acquired along the years and the success achieved in autonomous navigation in controlled environments, the team decided to deal with real road scenarios with a full size prototype of a Ford Escort Station Wagon of 1998 (Figure 1.2). Thus, the ATLASCAR project appear.

Figure 1.2: The ATLASCAR and some of its sensors.

The main goal of the ATLASCAR project is to develop new Advanced Driver Assistance Systems (ADAS). For that purpose, the car was equipped with a rich set of sensors dedicated mainly to the perception of the surrounding environment. Also, the car suffered some mechanical modifications in its interior to be able to perform some operations.

1.2 Problem Context

This work is part of the ATLASCAR project [1], carried out at the University of Aveiro, whose main purpose is the research and development of solutions on autonomous driving and Advanced Driver Assistance System (ADAS).

The car is equipped with several exteroceptive sensors, namely a stereo camera, a 3D LIDAR, a foveated vision unit and additional planar laser range finders. The planar lasers already installed on the car are two Sick LMS151, and the 3D laser is actually a Sick LMS200 in a rotating configuration. There is also a Hokuyo laser range finder to extract road profiles ahead of the car. Additionally, a new multi-layer laser (sick LD-MRS 400001) and two new Point Grey cameras are to be installed in the car and, finally, a SwissRanger 3D TOF camera is also occasionally used in some experiments and contexts.

With this number of sensors a calibration between them is extremely important to have a complete perception of the surrounding environment. Thus, the main objective of this project is to develop a method to calibrate all the sensors presents in the car.

1.3 Objectives

Currently the calibration on AtlasCar is made by hand. However, the main problem of this approach is not related with the translation between devices; the main problem are the rotations, which are extremely difficult to estimate accurately by manual means; the first problem starts by identifying the placement of the referential system on the devices and their axes directions; then, attaching the associated error of the measuring tool used to estimate the rotations between devices, easily a significant error is obtained. A small error in the rotation provides a larger error when compared with errors related with translation. Thus, the main objectives of this project are the following:

- Create an algorithm for the detection of the target with the laser sensors;
- Create an algorithm to calibrate the laser sensors;
- Implement a method to detect the target with the two cameras;
- Include the cameras on the calibration process;
- Tests and evaluation of the calibration method.

1.4 Related Work

Over the past several years, a number of proposed solutions for the calibration of a set of sensors (LIDAR and VISON) with respect to a global frame were developed because of its relevance for autonomous vehicles and robotic applications. The majority of the existent solutions have as common base of finding correspondences between different devices, being differentiated by the means to obtain those correspondences and how they are related.

In [2] a method to estimate the motion of a camera-laser fusion system was developed, which consists of a 2D laser sensor and multiple cameras designed to reconstruct a 3D structure of outdoor environment by capturing data while it passes through the target

area. The laser points are projected onto the images using the Kanade-Lucas-Tomasi tracker [3] and tracked to other frames to be used as 3D-2D correspondences using a three-point method approach. However, the generalized three-point algorithms (G3P) [4], has a limitation: it is not able to estimate the motion of the system if three points are collinear, which happens frequently when the laser sensor scans a large plane such as walls or ground, as illustrated on Figure 1.3. To prevent that situation the authors developed a solution based in the alternative algorithm that Bock et al. [5] presented, called laser three-point (L3P). To estimate the motion between consecutive frames this algorithm uses laser points from both frames; the union set of the lasers from both frames is expected not to be collinear if the system moves while capturing data. Figure 1.4 illustrates how the algorithm works, where the points Q1 and Q2 scanned at frame 1 and Q3 scanned at frame 2 are projected onto the corresponding images and tracked to the other frames. The angles between rays ($\theta 1$, $\theta 2$, $\theta 3$, $\phi 1$, $\phi 2$, $\phi 3$) and the distance to the points at the scanned frame (L_1, L_2, L_3) are known. The unknown variables are the distances to the points (l_1, l_2, l_3) at the tracked frames. Relating those variables, it is obtained three equations that can be computed by solving a four-degree polynomial equation.

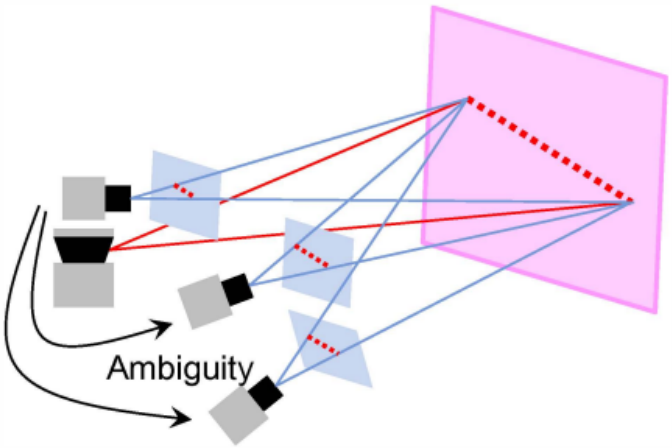


Figure 1.3: Limitation of scanning a plane with a 2D laser sensor on the typical three-point algorithms because all laser points are collinear. [2]

The authors presented the generalized laser three-point (GL3P) algorithm, which is an adaption of the L3P for a multiple camera configuration. Similarly to the L3P, two points and other point are selected from different frames. Then the problem is solved by transforming two points (selected from the same frame) on two rays and apply the inverse transformation to the third point (selected from the other frame).

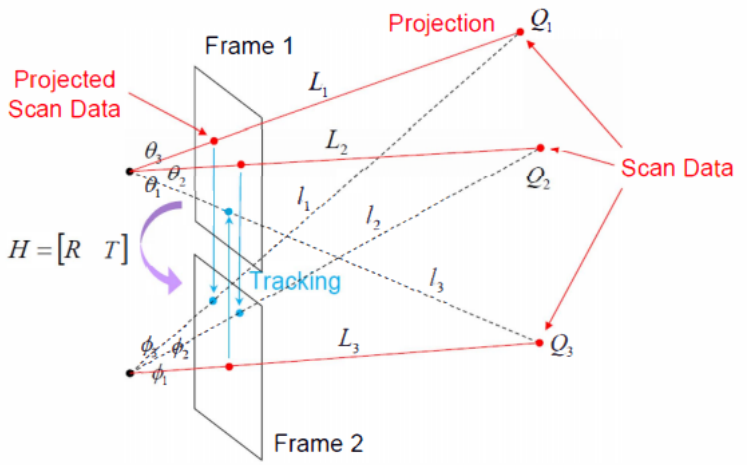


Figure 1.4: Laser three-point algorithm: Two points Q_1 and Q_2 are selected from the frame 1 and tracked to the frame 2. The other point Q_3 is selected from the frame 2 and tracked to the frame 1. [2]

In [6] an approach to solve the extrinsic calibration between a camera and a multi-layer laser range finder was presented. This method takes advantage of photographic and laser range data by using a circle-based calibration target. The circle-based calibration target is a rigid plane with a printed black ring. The inner circle is a plane perforation. The configuration is illustrated on Figure 1.5.

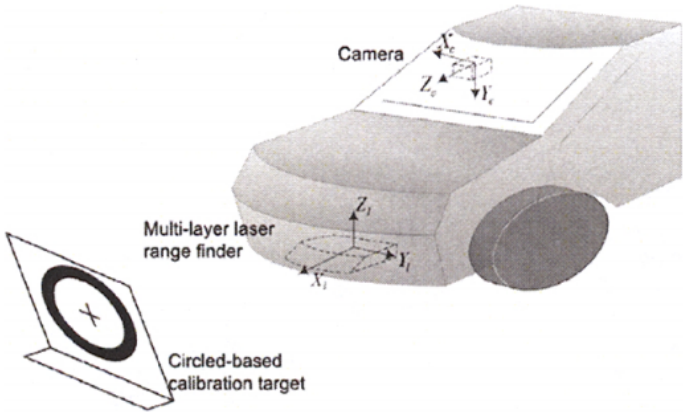


Figure 1.5: Overview of the proposed calibration as detailed in [6].

The proposed method consist of estimating different poses of the calibration object detected simultaneously by the camera and the multi-layer LIDAR, resulting in a set of point correspondences between frames (circle centers of each pose), which will be used to determine their relationships. Each pose of the calibration target is parameterized by 3D coordinates of the circle center and the normal vector of its plane.

After extracting the set of correspondences between camera and laser, the LIDAR-to-camera calibration is processed in two steps. First an initial guess from a linear solution is obtained, using a well-known closed-form solution developed by Arun et al. [7]; this

solution consists of determining the relationship between the two coordinate frames using sets of corresponding features through the singular value decomposition (SVD). And for last, the resulting transformation of the previous step is refined by minimizing a point-to-point error measure, similar to the Iterative Closest Point (ICP) algorithm.

In [8] is presented an on-line approach to recover the geometric transformation of two LIDAR lasers relatively to the vehicle. Although, this method make several assumptions; first, it is assumed that the vehicle platform has the ability to estimate its position and orientation with respect to a fixed world frame; second, an initial calibration estimation for each LIDAR is available; finally, it is also assumed that the LIDAR systems are capable of simultaneous measuring range and remission (reflectivity) γ values of features in the environment.

Taking in account those assumptions, a scene is prepared with hand placed retro-reflective tape upon poles along a calibration loop. Over that calibration loop the targets are automatically segmented from the background using a simple threshold ($\gamma > \gamma_{min}$) on the LIDAR remission measurements. The result of the segmentation are illustrated on Figure 1.6, where a sample landmark pole with two retroreflective marks is shown on the left, and on the right the same pole viewed by a scanning LIDAR with the location of the segmented targets through the threshold operation in red. After that, the segmented targets are reduced to a point, which will be used as a correspondence across multiple calibration loops. Finally, the extrinsic parameters of the laser in relation to the vehicle are estimated using the recovered point correspondences through a second-order cone programming (SOCP) [9].

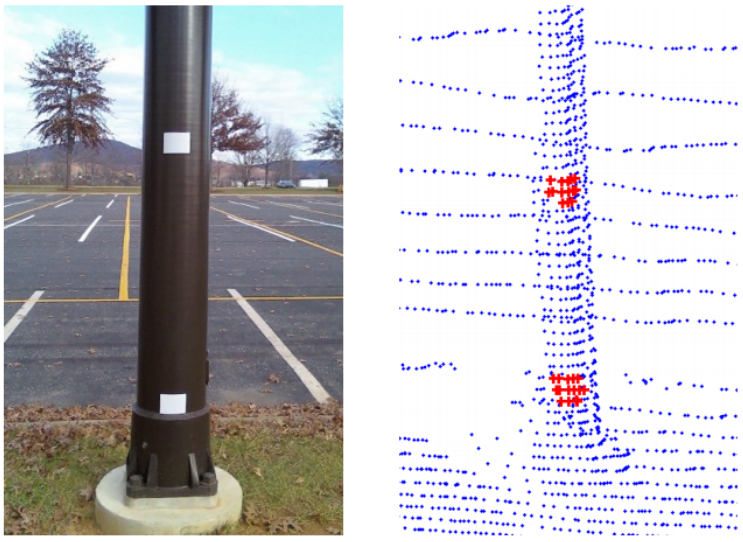


Figure 1.6: A landmark placed in a pole on the left, and the resulting segmentation of the LIDAR scan on the right. [8]

In [10] is presented a method for solving the extrinsic calibration between a camera and a multi-layer laser scanner by detecting a planar triangle board as illustrated on Figure 1.7. This method is based in three main steps:

- Acquiring synchronized data from the laser and camera;

- Select regions of interest and estimate the position of the calibration targets with respect to the camera and laser frames, in which the target poses are computed in 2D and 3D respectively;
- An optimization procedure based on geometric minimization that involves image and laser correspondences simultaneous by fitting virtual points O_t.

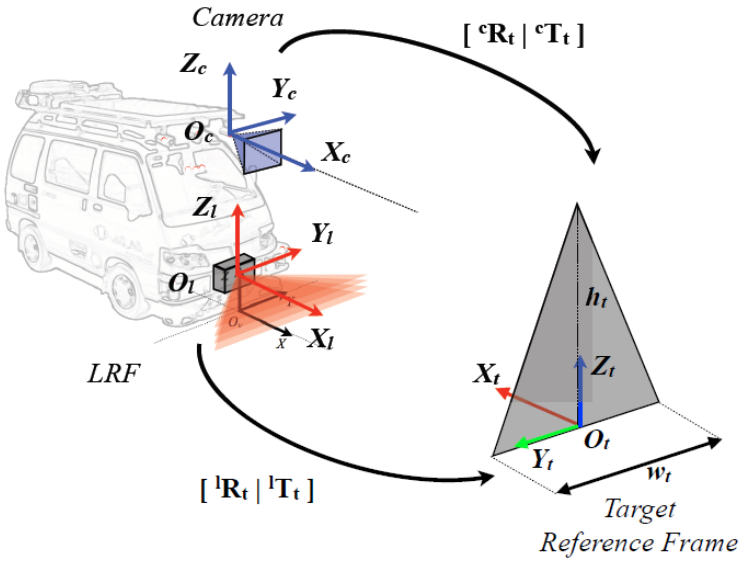


Figure 1.7: Overview of the proposed strategy in [10].

For the target detection in the laser frame, the scan is clustered, and only opportune planar thin objects will be kept. Next, RANSAC fitting is applied to all potential targets T_j to generate a target plane represented by a centroid and a normal vector model. Then, all the points owned by T_j are projected in that plane , which will be used to estimate the position of the calibration target with respect to the laser and calculate the coordinates of the virtual point.

To detect the target in camera frame, two procedures based on monocular view and stereo images were described. In mono images firstly a preprocessing stage involves sequentially color filtering and edges extraction that are applied on image, and Hough transformation is used to extract straight lines from it. After the preprocessing stage, parametric equations of linear components are computed and applied on image. The last stage concerns in matching between lines and edges previously obtained, where the lines that matches with the higher amount of white pixels are extracted with respect to a specific threshold. Finally, in this set of lines, each triangle is composed as a line triplet by matching couple of segment vertex, whether all vertex couples differ each other by less than a Manhattan threshold, and an intersection point of the lines will be computed. Only when three valid intersection points are obtained, a triangular shape is detected. In the case of stereo images the procedure is similar to the laser procedure due the same range information, however a different error distribution is present and must be consider in model extraction.

In order to compute the extrinsic parameters, the points in laser are projected in the image, which knowing the intrinsic parameters of the camera, the extrinsic transfor-

mation between the laser and the camera can be formulated as a non-linear problem to find the optimal solution of the rotation vector and translation vector. This non-linear optimization is solved by using the Levenberg-Marquardt algorithm.

A more recent work [11] presents a solution for the extrinsic calibration of 2D LIDAR lasers, which is based on the observation of perpendicular planes. These planes can be already present in the environment (something common in structured scenes), or constitute a calibration pattern built for the purpose. The calibration is constrained by imposing co-planarity and perpendicularity constraints on the line segments extracted by the different laser scanners. A rough approximation of the sensors relative poses must be provided. This method can be used to calibrate any set of rigidly joined lasers, where there are at least two sensors with non-parallel scanning planes. In summary, the necessary conditions for the method works are:

- An approximation of the sensors relative poses must be known;
- The sensors rig observes at least two perpendicular planes from two different viewing directions;
- Not all the sensors scan parallel planes, that means that the measurement planes of at least two sensors must intersect.

This proposed method is based in establishing geometric constraints from simultaneous observations of pairs of perpendicular planes (see Figure 1.8). Then, the geometric constraints are inferred from:

- the co-planarity of the observed line segments lying on each face of the corner
- the perpendicularity of both planar surfaces.

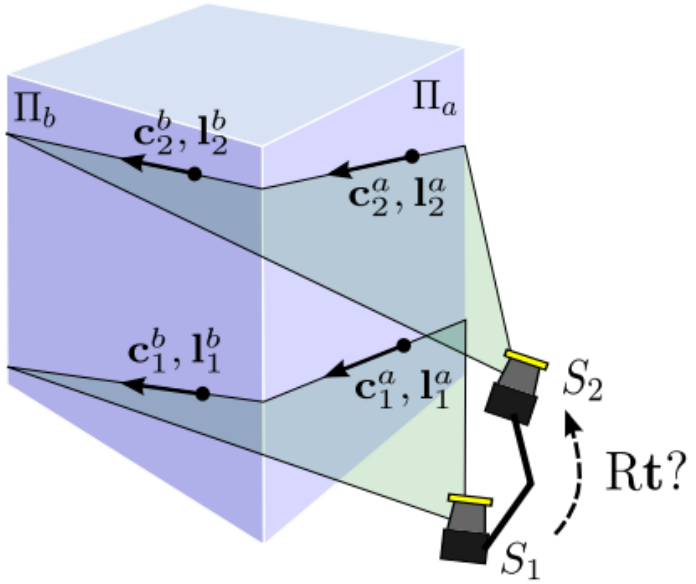


Figure 1.8: Observation of a corner structure by a rig with two LIDARs. [11]

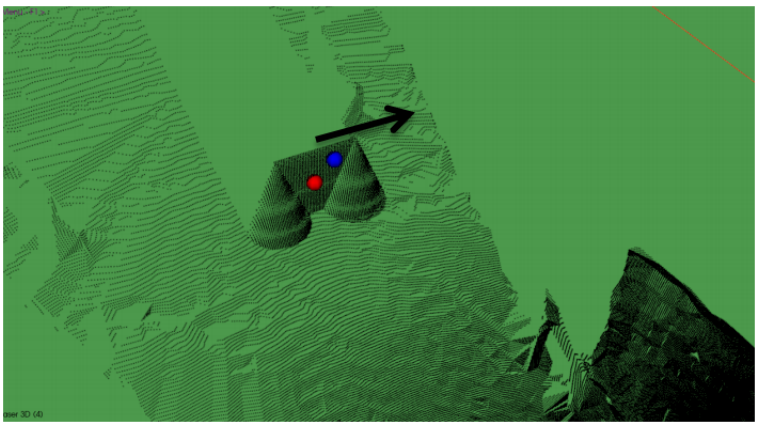


Figure 1.9: Calibration object detection in 3D data. [12]

In summary this method is divided in the following steps: gather observations, segment lines, generate corner candidates, selection of the corner candidates and compute calibration. The corner candidates are selected through co-planarity and orthogonality constraints.

To select the corner observations (COs), a number of observations are taken from different poses of the sensor rig, which applying RANSAC method the inconsistent corner candidates are excluded. For that, a candidate extrinsic calibration is calculated from a minimum set of randomly selected corner candidates. Then, the number of corner candidates consistent with such calibration is evaluated. This process is repeated iteratively by searching for a maximum consensus of corner candidates. The result of this process is a larger set of consistent candidates, which are the the COs from which the calibration is computed. The extrinsic calibration is calculated relatively to one of the sensors using a method of maximum likelihood estimation (MLE).

Finally, the work developed by Miguel Oliveira [12] presented an approach to calibrate 3D/2D lasers, where the 2D laser are calibrated with respect to the 3D laser frame. This method places a static calibration object in several positions in the acquired scene. The calibration object has two cones with an intermediate plane to keep the cones at a fixed distance. The cone geometry, due its geometric features, make possible the determination of the position and orientation of the 2D laser footprints regarding the 3D point cloud, since it removes the ambiguity in height. The combination with the second cone provides enough information for the orientation around the vertical axis.

The calibration process requires an initial approximation position of the calibration object in the 3D point cloud. User need a point in the center of the calibration object (red sphere on Figure 1.9) and an additional point in the plane (blue sphere on Figure 1.9), which is between the two cones. Those points define approximately the center and orientation around the vertical axis of the calibration object with respect to the 3D laser. The other rotations are not considered since it is assumed that the calibration object is in a flat surface. Then, once the dimensions of the object are known, the 3D date is filtered using a bounding box and the points that belong to the object are extracted. Next, a virtual calibration object is fit to the set of points extracted using the Iterative Closest Point (ICP) [13], giving the transformation matrix of the calibration object with respect to the 3D data frame.

With the detection of the calibration object in the 3D data, it is necessary to do the same with in 2D data, in order to compute the transformation between the 2D laser and the 3D laser. This detection is done by taking advantage of the geometry associated to the cones, in other words, the calibration target produces two ellipses in 2D data, which are used to identify the calibration object. The process of ellipse detection needs the user to select two points per ellipse as shown on Figure 1.10. Then, using a minimum square errors, is found the ellipse that best fit the extracted points. An additional step is used to get the angle between the calibration object and the ellipse's major axis. Then, the ellipse is fit into the cone, where the only indetermination is related with the rotation angle around the vertical axis of the cone, which is solved by relating the ellipse with a cone as shown in Figure 1.11.

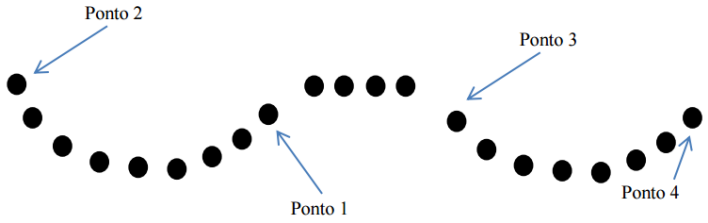


Figure 1.10: Points selection for ellipse reconstruction. [12]

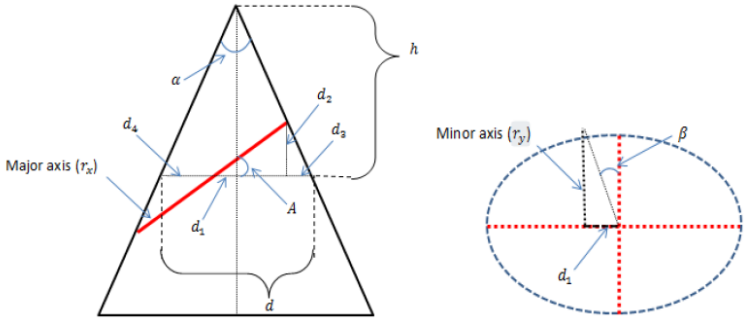


Figure 1.11: On the left a cone's cut along the major axis; on the right a projection of the ellipse. [12]

Finally, from the analytical model of each ellipse, four points are computed in the intersection of the ellipse with the major and minor axes, resulting in eight points from the fitting of the ellipses to 2D laser data. Also, another eight points are obtained from the fitting of the ellipses to the cones of the calibration object. Since this process is done for each ellipse, each calibration provide eigth points in the 3D laser cloud. The transformation matrix that relates the 2D laser with the 3D laser is obtained with functionalities offered by VTK [14], which transforms the eight points from the 2D laser data into the eight points of the 3D laser data.

Chapter 2

Software and Hardware Description

This chapter presents the description of the software and hardware used in this master thesis. In terms of software OpenCV, PCL and ROS were used. Relatively to the hardware a Sick LMS151 already installed in the car, the new multi-layer laser Sick LD-MRS400001, a SwissRanger SR4000 and two Point Grey cameras were used. Despite the SwissRanger not being part of the set of sensors typically on board a car for the autonomous driving, it provides a different type of data to be processed in comparison with the other two. Thus, it was possible to prove the versatility of the studied calibration method regarding the type of sensors that it supports.

2.1 Hardware

2.1.1 Sick LMS151

The Sick LMS151 (Figure 2.1) is an outdoor 2D laser rangefinder commonly used in robotics applications and for obstacle detection in autonomous driving. The LMS151 is characterized by being small and easy to mount, and it can measure distances even through glass and dust due its double pulse evaluation. The most relevant properties of the laser Sick LMS151 are presented on the Table 2.1.

Figure 2.1: Sick LMS151 laser rangefinder.

Table 2.1: Properties of the laser Sick LMS151

Operating range	0.5 m - 50 m
Max. range with 10 % reflectivity	18 m
Scanning angle	270°
Angular resolution	0.5°/0.25°
Scanning frequency	25 Hz/50 Hz
Systematic error	±30 mm
Statistical error	±12 mm
Laser protection class	Laser class 1
Operating temperature range	-30 °C to +50 °C
Dimensions (W x H x D)	162 mm × 102 mm × 106 mm
Total weight (without cables)	Approximately 1.1 kg
Supply voltage	10.8 to 30 V
Data interfaces	Ethernet 100 Mbit TCP/IP; RS 232; CAN (for the connection of an I/O module)

Data is received from the sensor as a set of points in polar coordinates. This set is sorted from starting angle ($\theta = -45°$) to the stopping angle ($\theta = 225°$). Considering the number of measurements per scan n, the ranges $\{r_1, ..., r_n\}$ and the angles between the measured point and the y axis $\{\theta_1, ..., \theta_n\}$. The cartesian coordinates (x_n, y_n) for each point of the scan are calculated as follow:

$$\begin{cases} x_n = r_n \cos\theta_n \\ y_n = r_n \sin\theta_n \end{cases} \tag{2.1}$$

2.1.2 Sick LD-MRS400001

The Sick LD-MRS40001 is a multi-layer laser scanner (Figure 2.2) commonly used for automation with object tracking or field evaluation. This laser has four simultaneous scanning layers, which allows angle compensation by means of the different vertical angles for each layer. Taking as example the application that it will have, the LD-MRS detects the object reliably even when a car brakes or accelerate.

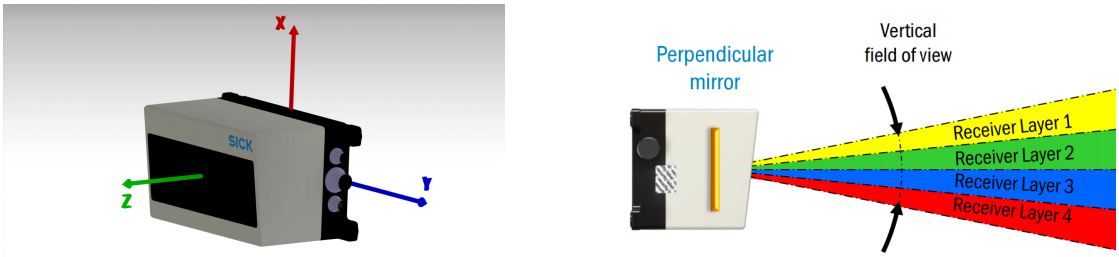
This laser can also detect three consecutive echo pulses per measurement and per plane. Due this capacity, the LD-MRS can differentiate measurements of interference such as rain, snow or dust. All those characteristics give to the LD-MRS the ability to withstand outdoor conditions, even under poor environmental conditions. More of its characteristics are presented on Table 2.2.

Figure 2.2: Sick LD-MRS400001 multi-layer laser.

Table 2.2: Properties of the laser Sick LD-MRS400001

Operating range	0.5 m - 250 m
Max. range with 10 % reflectivity	50 m
Scan planes	4, vertical aperture angle over 4 planes: 3.2°
Aperture angle	85°, operating angle with 4 measurement layers, 25° work area expansion with 2 measurement layers (total 110°)
Angular resolution	0.125°/0.25°/0.50°
Scanning frequency	12.5 Hz/25 Hz/50 Hz
Statistical error	±100 mm
Echoes per individual measurement	3
Laser protection class	Laser class 1
Operating temperature range	-40 °C to +70 °C
Dimensions (W x H x D)	88 mm × 164.5 mm × 93.2 mm (including fastening tabs)
Total weight (without cables)	Approximately 1 kg
Supply voltage	9 to 27 V DC,
Data interfaces	Ethernet 100 Mbit TCP/IP; RS 232; CAN (for the connection of an I/O module)

Like the LMS151, the data received from the LD-MRS is in polar coordinates; although, it has four layers with a vertical aperture between layers of 0.8°, which provides 3D coordinates. Figure 2.3 illustrates the configuration of the four layers and the configuration of the coordinate system used.



(a) Coordinate system.

(b) Configuration of the four layers.

Figure 2.3: Sick LD-MRS400001 multy-layer configuration.

Taking as example the Figure 2.4 that illustrates a measured point by the laser; lets consider i the number of the layer, n the number of the measurement in the scan of each layer, the range $\{r_{1_i}, ..., r_{n_i}\}$, the angles between the measured point $\{\theta_{1_i}, ..., \theta_{n_i}\}$ and the plane XY and the angles between the layer and the plane YZ $\{\alpha_1, ..., \alpha_i\}$. The Cartesian coordinates are calculated as follow:

$$\begin{cases} x_{n_i} = r_{n_i} \cos\theta_{n_i} \\ y_{n_i} = r_{n_i} \sin\theta_{n_i} \\ z_{n_i} = r_{n_i} \sin\alpha_i \end{cases} \tag{2.2}$$

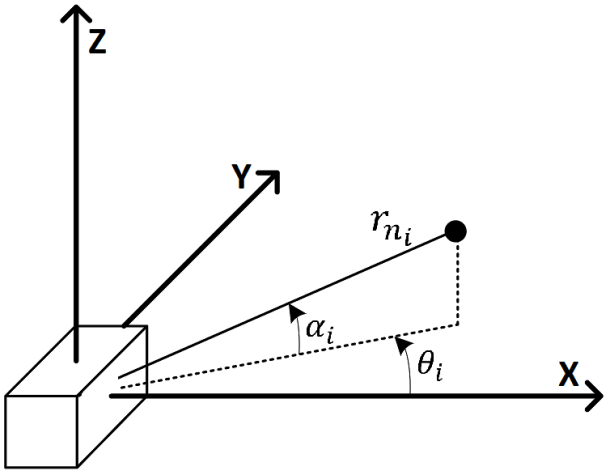


Figure 2.4: Illustration of a measured point by the Sick LD-MRS.

2.1.3 SwissRanger

The SwissRanger SR40000 (Figure 2.5) is a Time-of-flight (TOF) 3D camera that provides high-resolution 3D image data in real time. The TOF principle uses the emitted infrared light from the camera's internal lighting source, which is reflected by the surrounding objects in the scene and travels back to the camera where its precise time of arrival is measured independently by each sensor pixel. Also, the SR40000 has a programmable modulation frequency (20/30/31 MHz) that allows the use of simultaneous measurements of three cameras without interference.

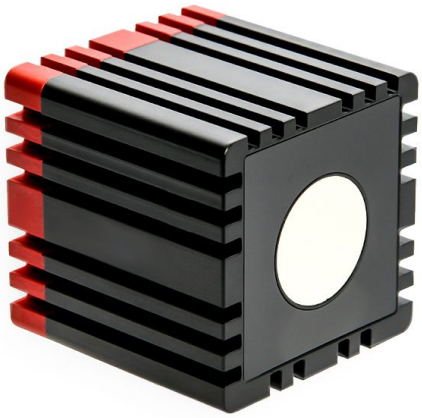


Figure 2.5: SwissRanger SR40000 3D camera.

This type of camera is designed for indoor environments and has an optical filter in front of the sensor to allow only wavelengths near of the emitted by the camera pass into the camera lens. Due the characteristics of the SR40000, it is specially attractive for indoor robotic applications. Some of its most important characteristics are illustrated on Table 2.3.

Table 2.3: Properties of the laser SwissRanger SR40000

Detection range	0.1 m - 5.0 m
Modulation frequency	29/30/31 MHz
Angular resolution	0.24?
Absolute accuracy	±10 mm
Maximum frame rate	50 FPS
Pixel array size	176 (h) x 144 (v)
Field of view	43.6° (h) x 34.6° (v)
Operating temperature	+10 °C to +50 °C
Storage temperature	-20 °C to +70 °C
Dimensions (W x H x D)	65 mm × 65 mm × 68 mm
Weight	470 g
Electrical power requirements	12 V (-2%; +10%), maximum 1.0 A, (typical 0.8 A)
Data interface	USB

The SR40000 does not require data treatment since the data provided is already in cartesian coordinates.

2.1.4 Point Grey Camera

The cameras used are a Flea3 Gigabit Ethernet, more specifically the model FL3-GE-28S4-C (Figure 2.6). This camera use a Sony ICX687 EXview HAD CCD II image sensor

to deliver high resolution, high quality images in a compact and low cost package. Also, is can runs at 15 FPS at full 1928 x 1448 resolution, and even faster using smaller regions of interest, measuring just 29 x 29 x 30 mm and a weight of only 38 g. This combination of resolution, sensitivity and size make these models ideal for applications that require low cost and high-speed camera with compact design, such as: in factory automation and machine vision; medical and life science applications; and intelligent transportation systems.

Despite all this characteristics, the Flea3 GigE cameras offers other advantageous features, like an 8-pin opto-isolated GPIO for industrial triggering and strobe output; 1 MB non-volatile flash memory for user data storage; and on-camera frame buffer for retransmitting images. Some of its properties are presented on Table 2.4.

Figure 2.6: Point Grey camera FL3-GE-28S4-C.

Table 2.4: Properties of the laser Point Grey camera FL3-GE-28S4-C

Sensor size	1/1.8"
Interface	GigE
Shutter	CCD
Max resolution	1928 × 1448
Max frame rate	15 FPS
Pixel size	3.69 µm
Megapixels	2.8 MP
Chroma	color
Power Requirements	12-24 V
Dimensions (W x H x D)	29 mm × 29 mm × 30 mm
Weight	38 g (without optics)
Temperature (Operating)	0° to 45°C
Temperature (Storage)	-30° to 60°C
Humidity (Operating)	20 to 80% (no condensation)
Humidity (Storage)	20 to 95% (no condensation)

2.2 Software

2.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) [15] is an open source computer vision and a learning software library written in C and C++ and runs under Linux, Windows and Mac OS X. OpenCV was designed for computational efficiency and with strong focus on real-time applications. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quick. To achieve that OpenCV library contains more than 2500 optimized algorithms that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. The OpenCV's library is well documented, and due the large collaborative user community is in constant growth.

For this work, OpenCV's will have special interest in stereo vision as will be further presented.

2.2.2 PCL

The Point Cloud Library (PCL) [16] is a large scale, open project for 2D/3D image and point cloud processing. PCL is a cross-platform written in C++, and has been compiled and developed on Linux, MacOS, Windows, and Android/iOS.

The PCL incorporates multiple processing algorithms that operate on point cloud data including: filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. Some of these algorithms can be used to filter outliers from noisy data, stich 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute description to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them. Each set of algorithms is defined via base classes that attempt to integrate all the common functionality used throughout the entire pipeline, thus keeping the implementations of the actual algorithms compact and clean. The basic interface for such a processing pipeline in PCL is:

- creat the processing object (e.g., filter, feature estimation, sementation);
- use *setInputCloud* to pass the input point cloud dataset to the processing module;
- set some parameters;
- call *compute* (or *filter*, *segment*, etc) to get the ouput.

2.2.3 ROS

ROS [17] is an open-source development environment specifically for applications in robotics, usually used in large projects that, due to its modular architecture, it reduces the projects complexity by splitting a large project into smaller modules, each with a specific application. Also, those modules can be used for other applications and not only in a single project. It provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes and package management.

One of the main goals of ROS is to make easy the communication between hardware and software using a peer-to-peer topology, where nodes can communicate directly with another without the need of a central server. However, it's not typically feasible for each node to be directly connected to other node, so, instead the nodes connect to some subset of nodes. Information can be published to a network in a broadcast fashion making this information available for all nodes, but it's not possible to directly address it to a node. This topology permits to run a great number of executables in parallel that need to be able to exchange data synchronously or asynchronously. A system built using ROS consists of a number of processes, potentially on a number of different hosts, connected at runtime in a peer-to-peer topology. The concepts of the implementation of the peer-to-peer topology are: nodes; master; parameters server; messages; topics; services; and bags.

The peer-to-peer topology requires some sort of lookup mechanism to allow processes to find each other at runtime. We call this the name service, or master. It stores topics and services registration for ROS nodes. Nodes communicate with the master to report their registration information. As these nodes communicate with the master, they can receive information about other registered nodes and make connections as appropriate. The master will also make call backs to these nodes when this registration information changes, which allows nodes to dynamically create connections as new nodes are run.

Nodes connect to other nodes directly, the master only provides the lookup information. The connection between node is performed using a publish/subscribe messaging model into topics. These messages are a simple data structure already predefined by ROS. In resume, a node that subscribe to a topic will request connection from the node that publish into that topic and establish a connection. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data, subscribe to the relevant topic. This is important for large projects because it's possible to have several topics being published and subscribed for multiple nodes. Figure 2.7 shows how this architecture works.

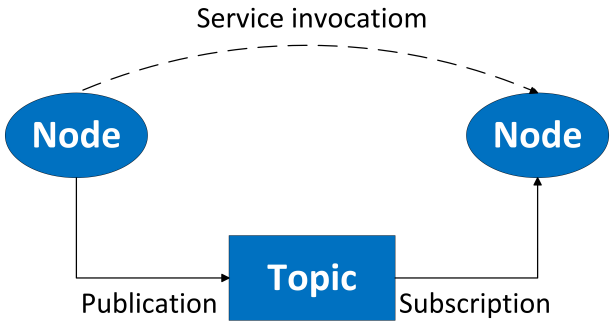


Figure 2.7: ROS based architecture: nodes are programs that exchange data through messages/topic subscription and services.

Chapter 3

Methodology for Sensors Calibration

This chapter presents the methodology for the sensors calibration. The main goal of calibration is from a set of sensors, obtain the extrinsic parameters that describes the geometric transformation (rotation and translation) between two coordinate systems. This calibration process is important when a system has more then one sensor; if the data from each sensor is relative to a global coordinate frame, a more complete information about the covered scene is obtained by the sensors, which can for example help in object detection, and prevent some false object detections. Figure 3.1 shows two lasers measuring a ball from two distinct positions, and Figure 3.2 shows the difference between uncalibrated and calibrated sensors with respect to a reference frame.

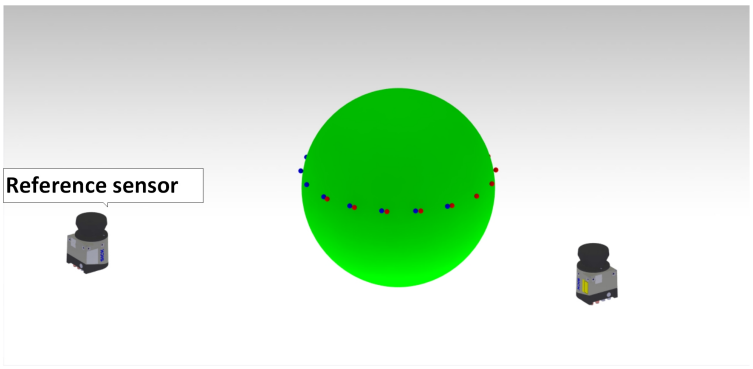


Figure 3.1: Two sensors measuring a ball.

Uncalibrated (overlapped sensors). Calibrated.

Figure 3.2: Difference between uncalibrated and calibrated sensors.

For this work the set of sensors is composed by a Sick LD-MRS400001, a Sick LMS151, a SwissRanger and two Point Grey cameras. The idea to calibrate the sensors is of finding correspondences between sensors using a ball as target, allowing its center detection by all sensors. Given the different set of sensors (2D LIDAR, 3D data, and images), the process is different for each type. For LIDAR data, mathematical properties of the sphere and algorithms provided by PCL are used. On the case of vision, two methods using OpenCV algorithms are used. One is to find the circle that fits the ball on image; the second is to use the two cameras as a stereo system. The choice turned out to be the stereo system, using the two cameras mounted on a stereo rig as shown on Figure 3.3.

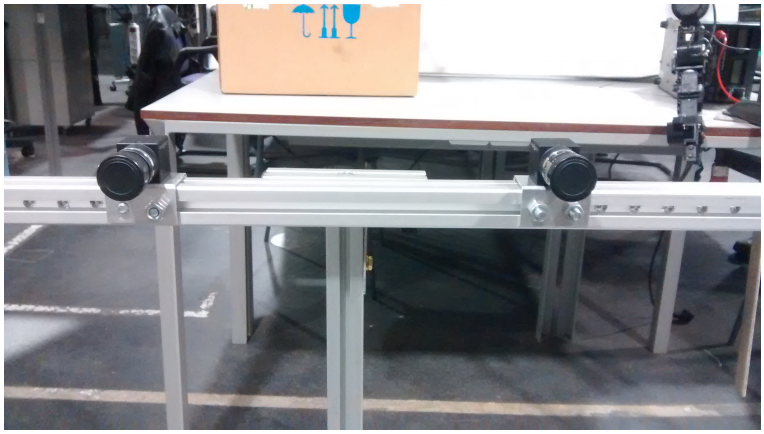


Figure 3.3: The two cameras mounted on a stereo rig.

The methodology and some preliminary results of the LIDAR sensors calibration was already presented in [18]. All the processes and choices made are detailed on the next sections.

3.1 Calibration Procedure

Calibration consists in estimating the rigid body transformation between all the sensors with respect to a reference one; the proposed solution is to use a ball as calibration target; the only restriction of the calibration target is about its size (diameter). The size of the calibration target is related to the angular resolution of the lasers used; after some empirical experiments, it was observed that, in order to obtain feasible results, the ball must have a diameter large enough for the sensors to have at least 8 measurements on the target at 5 m, which means that for finer angular resolutions it is possible to use a ball with a smaller diameter.

The approach used to obtain the calibration between all the devices is achieved in three stages. First, each sensor must detect the center of the ball; then, the ball is placed in motion in front of the sensors allowing them to detect its center along successive positions, creating at the same time a point cloud for each sensor; the condition to consider a new point for each point cloud is that each new point is separated from the previous point some minimum distance. Finally, a sensor is chosen to be the reference frame and the remainder are calibrated relatively to this each one at a time by using an algorithm of the PCL [16]. Figure 3.4 illustrates the calibration approach, where several

position of the ball are taken (black points) by all sensors from the movement of the ball in front of them.

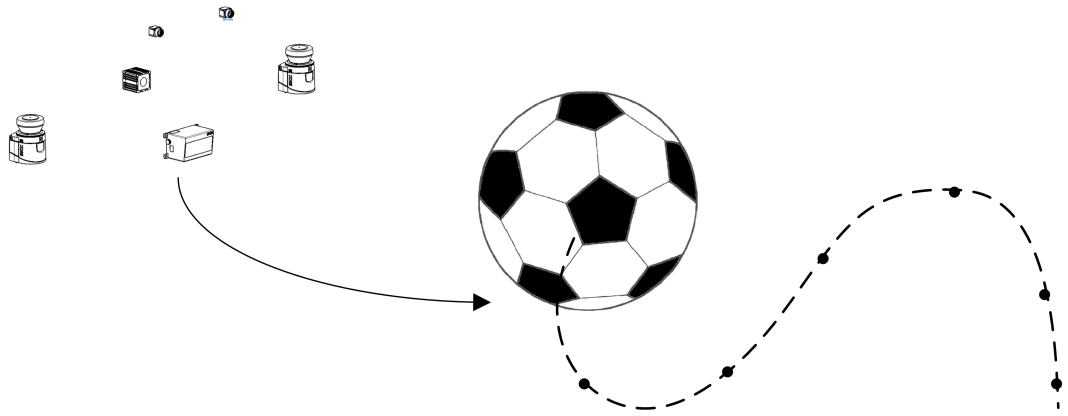


Figure 3.4: Overview of the calibration approach.

3.2 Calibration Target

The main reason to choose a ball as calibration target was mainly due its geometry. This object has the particularity that any planar section of the ball is a circle, a geometry well known and relatively easy to detect avoiding the necessity of detecting more complex geometries, having particular interest in 2D sensors. For the 3D sensors that have not the require of detecting circles, the mathematical model of a sphere as well as the circle is well known. Further will be explained the importance of those characteristics on the process of the ball center detection. Also, due the sphere form of the ball, its geometry is always the same independently of the point of view.

The ball used during the calibration is illustrated on Figure 3.5.

Figure 3.5: Real ball used as calibration target. The diameter is 107cm.

3.3 Sphere Center Detection on LIDAR Lasers

The method to find the center of the ball depends on the type of the range data. The method for 2D data is based in finding circular arcs, taking advantage on the particularity that any planar section of the ball is a circle. Thus, the process to detect the center of the ball is divided in the following sequence: segmentation of the laser scan, detection of circle and calculation of its properties (center coordinates and radius), and calculation of the center of the ball given the properties of the target (diameter). Figure 3.6 shows the diagram of the process.

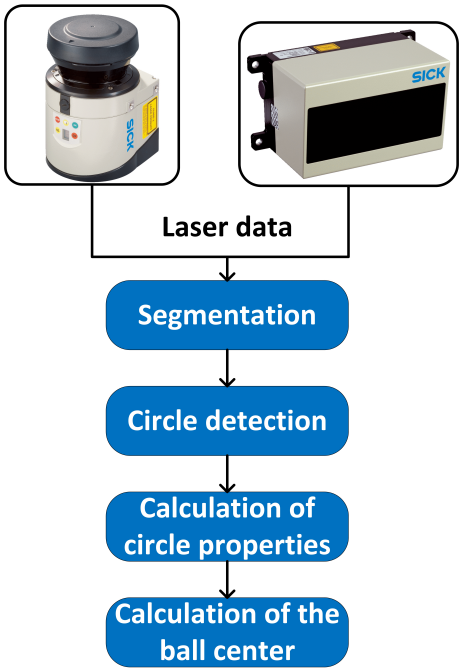


Figure 3.6: Overview of the steps to find the center of the ball.

It is important to mention that in 2D scans, due the symmetry of the ball, there is the ambiguity relatively to which of the hemispheres belongs the detected circle, since every section of the ball has a symmetric section relatively to the hemisphere, resulting in two solutions for the center of the ball (one above and another below the detected section). Consequently, some a priori information about the position of the sensor relatively to the ball is required. For the Sick LD-MRS the idea was to solve this problem taking advantage of its multi-layer technology, however, due its associated error it is not possible. After some tests with the Sick LD-MRS, it was verified that when all the laser scans were below the hemisphere of the ball, the lower scan should give the smallest circle diameter (since it is the smallest section of the ball), however, that's not always confirmed because of the sensor error. Thus, also a priori information about this laser is required.

3.3.1 Segmentation

Segmentation is a very important part of the calibration process; sets of points are generated with a probability of belonging to the ball, being then evaluated. The main

goal is to cluster the point cloud in sub sets of points which have high probability to belong to the same object through detected discontinuities in the laser data sequence, which are called break-points. A bad segmentation can compromise the rest of the process, which may cause problems to identify the ball.

In order to find the break-point, a Distance-based method is used to compute the Euclidean distance between two points, $D_E(p_i, p_{i+1})$, and if that distance is greater than a threshold D_{th}, then a break-point is detected.

Several methods are available to perform 2D point cloud segmentation. Based on the work from Coimbra [19], the Spatial Nearest Neighbour (SNN) was used since it appears the most consistent over different tested scenes.

Consider a full scan *Scan* as an ordered sequence of n measurement points ($\mathbf{p}$) in the form:

$$Scan = \{p_1, p_2, ..., p_n\}. \tag{3.1}$$

The raw data collected by the LIDAR is defined in a $\mathbb{R}^2$ space. These scanner points are commonly defined in polar coordinates (r_i, α_i) but also in cartesian coordinates (x_i, y_i), meaning:

$$\begin{cases} x_i = r_i \cos\alpha \\ y_i = r_i \sin\alpha \\ r_i = \sqrt{(x_i)^2 + (y_i)^2} \end{cases} \tag{3.2}$$

The two scan coordinates are illustrated more clearly on Figure 3.7.

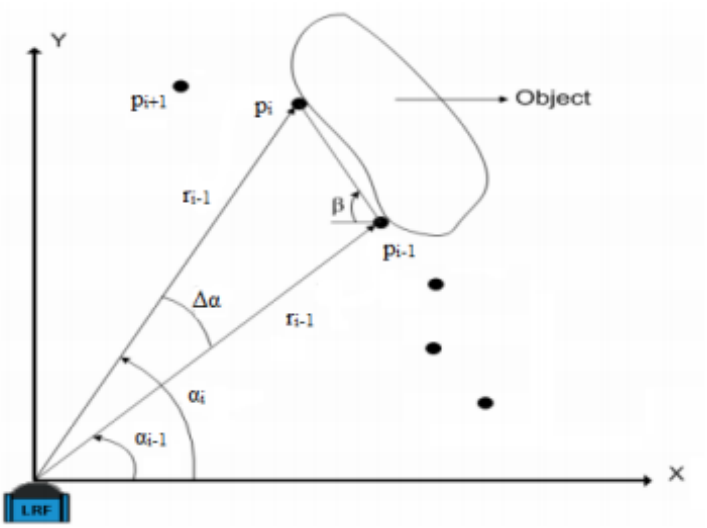


Figure 3.7: Geometrical representation of a hypothetical laser data. [19]

With the points defined in cartesian coordinates the Euclidean distance is calculated as follows:

$$D_{eu} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}. \tag{3.3}$$

The SNN is a recursive algorithm where the distance between a scanned point and all the other points that are not yet assigned to a cluster is computed. If that distance is smaller than a certain threshold the points are assigned to that cluster.

The algorithm can be explain as follows:

If p_i not yet assigned to a cluster **then**
Calculate the Euclidean distance D_{eu} to all the other unsigned points
if $D_{eu} < D_{th}$ **then**
Points will belong to that cluster
end
Go to the next unsigned point and repeat procedure until there are no more unsigned points
end

The only variable in this algorithm is the threshold value, so it is expected that for a higher D_{th} the result will be bigger clusters, and for a smaller D_{th} the result will be smaller clusters.

Since this method calculates the Euclidean distance to all the others points (not yet assigned to a cluster), and just not to its neighbour, it is possible that the point p_i and p_{i+3} belong to the same cluster, while the points p_{i+1} and p_{i+2} belong to another. Figure 3.8 illustrates a case when this situation occurs, where a smaller object is occluding a bigger one and the last keeps being one cluster.

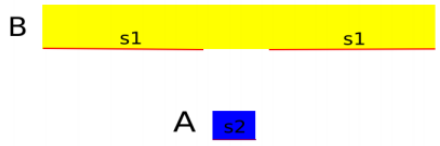


Figure 3.8: Segmentation Using the SNN, non-consecutive background points are assigned to the same segment (s1).

In conclusion this method can partially solve some occlusion related problems presenting better results than others. Figure 3.9 shows the result of the segmentation in a scan and the cluster related to the ball.

3.3.2 Circle Detection

The method used for circle detection is inspired on a work developed for line, arc/circle and leg detection from a laser scan data [20]. The circle detection makes use of a technique named Internal Angle Variance (IAV). This technique uses the trigonometric properties of the arcs: every point in an arc has congruent angles (angles that have the same amplitude) in respect to the extremes.

This property can be verified in Figure 3.10. Let P1 and P4 be the extremes of the arc, and P2 and P3 random points belong to the same arc. Then $(\angle P_1P_2P_4) = (\angle P_1P_3P_4)$ holds, because both angles measure one-half of $(\angle P_1OP_4)$.

The detection of circles involves calculating the mean of the aperture angle ($\bar{m}$) between the extreme points and the remaining points of a cluster, as well as the standard deviation (σ).To verify a positive detection at the beginning, values of standard deviation smaller than 8.6^{o} and values of mean aperture between 90° and 135° were used,

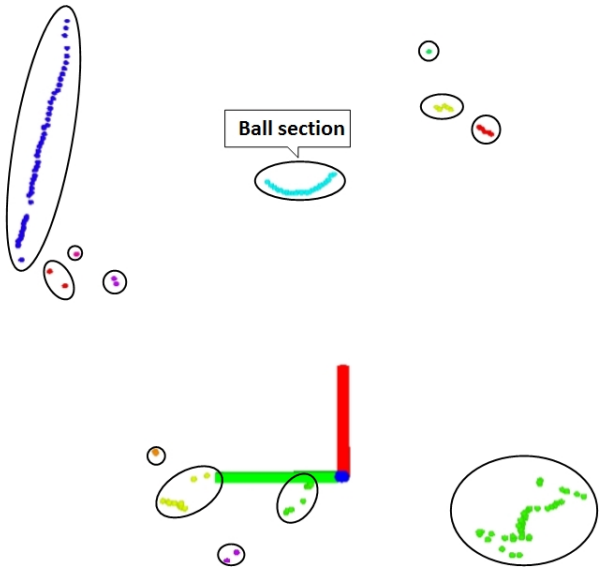


Figure 3.9: Result of the segmentation on sensor Sick LMS151.

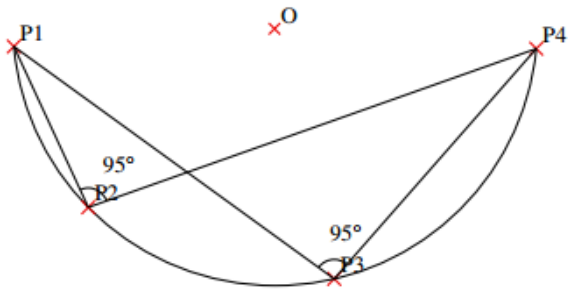


Figure 3.10: Congruent angles of points on an arc in respect to the extremes.

considering that the scan covers approximately half a circle. However, those values are dependent of two factors: the error associated to the sensors, and how much of the circle the scan covers. Thus, after analysing the results empirically, the values were adjusted to standard deviations values under 5º and values of mean aperture between 105° and 138° for the sick LMS151, and 10° and between 110° and 135°, respectively for the sick LD-MRS400001. These adjustments allowed to obtain the best results avoiding the detection of false circles.

Considering that a segment C has n points (P), say $C = \{P_1, P_2, ..., P_n\}$, the mean and standard deviation of the angle are calculated as follow:

$$\bar{m} = \frac{1}{n-2} \sum_{i=2}^{n-1} m(\angle P_1 P_i P_n) \tag{3.4}$$

$$\sigma = \sqrt{\frac{1}{n-2} \sum_{i=2}^{n-1} (m(\angle P_1 P_i P_n) - \bar{m})^2} \tag{3.5}$$

The confidence of the analyse increases with the following factors:

- The cluster has more than 8 points;
- For values of standard deviation below 5°;
- The average aperture angle is near of 90°, this means that the scan covers approximately half a circle.

3.3.3 Calculation of Circle Properties

The calculation of the center and radius of the circle uses the method of least squares to find the circle that best fits the points. Given a finite set of points in $\mathbb{R}^2$, say $\{(x_i, y_i)|0 \leq i < N\}$, first calculate their mean values by $\bar{x} = \frac{1}{N}\sum_i x_i$ and $\bar{y} = \frac{1}{N}\sum_i y_i$. Let $u_i = x_i - \bar{x}$, $v_i = y_i - \bar{y}$ for $0 \leq i < N$, and defining $S_u = \sum_i u_i$, $S_{uu} = \sum_i u_i^2$, etc. The problem is to solve first in (u, v) coordinates, and then transform back to (x, y).

Considering that the circle has center (u_c,v_c) and radius R, the main goal is to minimize $S = \sum_i g(u_i, v_i)^2$, where $g(u, v) = (u - u_c)^2 + (v - v_c)^2 - \alpha$, and $\alpha = R^2$. To do that, it is necessary to differentiate $S(\alpha, u_c, v_c)$, giving the following solutions:

$$u_c S_{uu} + v_c S_{uv} = \frac{1}{2}(S_{uuu} + S_{uvv}) \tag{3.6}$$

$$u_c S_{uv} + v_c S_{vv} = \frac{1}{2}(S_{vvv} + S_{vuu}) \tag{3.7}$$

Solving equations (3.6) and (3.7) simultaneously gives (u_c, v_c). Then the center (x_c, y_c) of the circle in the original coordinate system allows to write $(x_c, y_c) = (u_c, v_c) + (\bar{x}, \bar{y})$, and calculating α as follow:

$$\alpha = u_c^2 + v_c^2 + \frac{S_{uu} + S_{vv}}{N} \tag{3.8}$$

The result of the combination of circle detection with the properties of the circle is illustrated on figure 3.11.

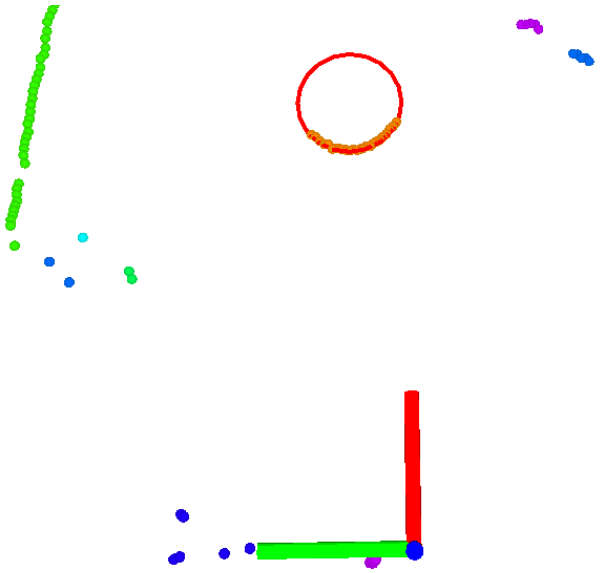


Figure 3.11: Result of the circle detection in real data from the sensor Sick LMS151.

3.3.4 Calculation of the Ball Center

After knowing all the properties of the circle, it is possible to calculate the center of the ball through trigonometric relations as shown on Figure 3.12, where R is the radius of the ball, R' the radius of the circle and d the distance between the center of the circle and the center of the ball, with $d = \sqrt{R^2 - R'^2}$.

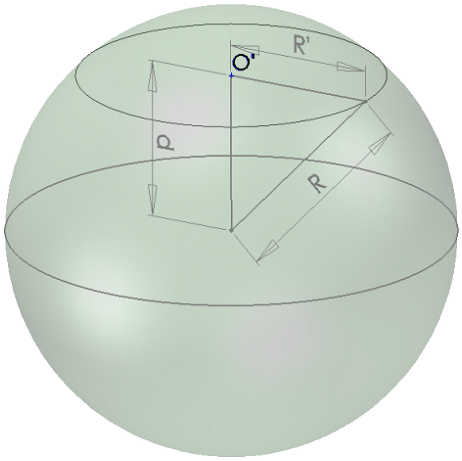


Figure 3.12: Example of the cross-section of the sphere at a distance d from the sphere's center.

Taking into account the ambiguity for the 2D lasers mentioned in section 3.3, and considering that the center of the circle has (x_c,y_c) coordinates, the coordinates of the center of the ball for this laser are defined as follows: $(X_c, Y_c, Z_c) = (x_c, y_c, \pm d)$. Figure 3.13 illustrates an example of the ball detection in a 2D scan.

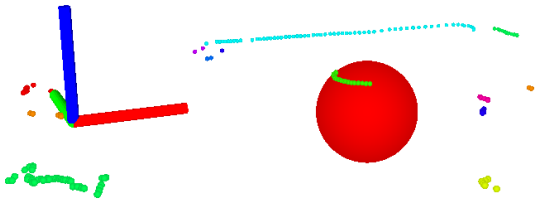


Figure 3.13: Detection of the ball in a 2D scan from the Sick LMS151.

In order to simplify the calculations in the 3D multi-layer laser, the coordinates of the circle center are transformed into the XY plane for each layer; then for each layer, the center of the ball is calculated precisely in the same way as for the 2D laser. After, the centers of the ball are transformed back into the respective original plane.

At this point, for each layer, the center of the ball was calculated, which means that there are as many centers as layers. Thus, statistically, the center of the ball is obtained by calculating the mean of all the centers. Considering that there are n layers and the different circles have $(x_{c_1}, y_{c_1}, z_{c_1}; ...; x_{c_n}, y_{c_n}, z_{c_n})$ coordinates, the center mean coordinates are defined as follows:

$$\begin{cases} X_c = \frac{1}{n}\sum_{i=1}^{n} x_{c_i} \\ Y_c = \frac{1}{n}\sum_{i=1}^{n} y_{c_i} \\ Z_c = \frac{1}{n}\sum_{i=1}^{n} z_{c_i} \end{cases} \tag{3.9}$$

Figure 3.14 shows an example of the detection of the ball by the multi-layer laser Sick LD-MRS.

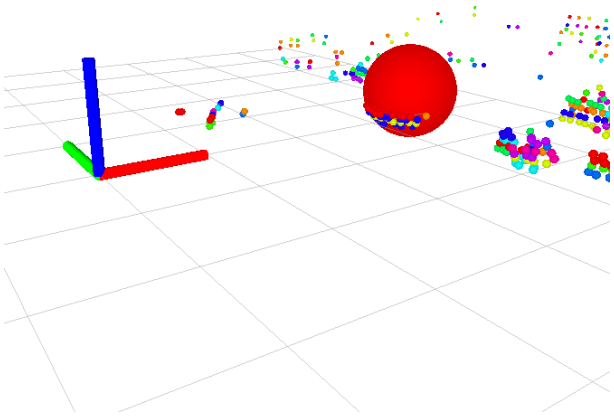


Figure 3.14: Detection of the ball on Sick LD-MRS.

3.4 Detection of Sphere Center in 3D Data

The ball recognition in 3D data, the case of the SwissRanger and stereo system, is achieved using the PCL segmentation capabilities, namely the sample consensus module, which accepts as input a point cloud and outputs the geometric model. In this case the model is a sphere, so the output produces the coordinates of the center of the ball and the radius.

The PCL implementation to compute the fitting of a sphere uses a Random Sample Consensus (RANSAC) [21] estimator method and a sphere model. The algorithm is an iterative method used to estimate parameters of a mathematical model from a set of data (3D point cloud from sensors) containing outliers. The RANSAC algorithm assumes that all the data includes of both inliers and outliers. Inliers are known as data points whose distribution fits some set of model parameters. On the other hand, RANSAC call outliers all those points that do not belong to the surface of a model. The goal of the RANSAC algorithm is to estimate which of the input points are inliers and which ones are outliers. The final result of the algorithm will be a set of estimated inliers and a set of estimated outliers. The RANSAC algorithm is composed of two phases that are repeated in a loop for each iteration.

The first phase is known as hypothesize phase. In this step, RANSAC follows a sampling strategy based on minimal sampling set of points (MSS). These points are randomly selected from the input data points, such that the shape parameters are computed by using only this minimal set.

In the second phase, which is known as testing phase, the quality of the MSS is evaluated. In order to carry out this task, the algorithm checks how well the model parameters, which were computed in the first phase, fit the rest of the points from the input data set. Each MSS is tested against the whole data points checking which points are closer to the candidate model. All the points which support the MSS configure the set of inliers for the fitting model, called consensus set.

The RANSAC algorithm keeps executing the loop until the probability of improving the consensus set drops below a given threshold. Consequently, after a certain number of iterations, the MSS with the best consensus set is chosen as the final model and all these points will be called estimated inliers. Figure 3.15 shows a point cloud received from the SwissRanger and result of the application of RANSAC algorithm on the detection of the ball.

3.5 Detection of the Center of the Ball with Cameras

The first approach to compute the center of the ball was using the Hough Transform provided by OpenCV. The Hough Transform is a method used for extracting features of specified shape in an image. Since the calibration target is a ball, Hough Circle Transform was used to find the circle that best fits the ball on image, returning its parameters (radius and center). Then, knowing the intrinsic parameters of the camera, those parameters would be used to relate the radius of the ball (which is known) with the radius and center of the circle in pixels, allowing the computation of the homogeneous coordinates of the ball with respect to the camera frame.

The Hough Circle Transform works in a manner roughly analogous to the Hough line transform, since it can be computed like the line detection case. The difference is

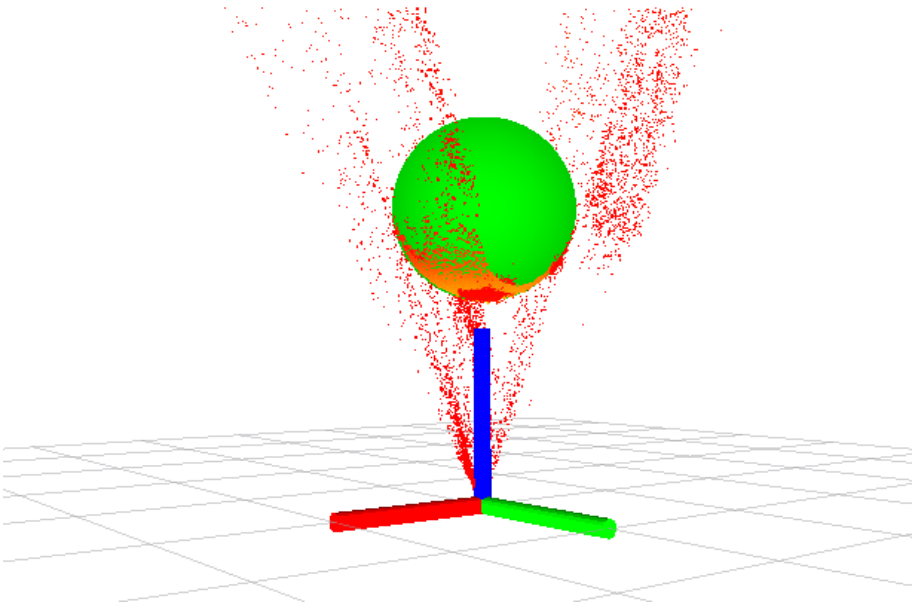


Figure 3.15: Ball detection in the SwissRanger point cloud.

that a line is defined by two parameters, and a circle is defined by three parameters (x_{center}, y_{center}, r), so, the accumulator plane used by Hough line transformation would have to be replaced by an accumulator with three dimensions; one for x_{center}, one for y_{center}, and another for the circle radius r. This would lead to considerable computational demand. To handle the dimensionality problem the OpenCV implementation uses the Hough gradient method [22]. This approach makes clever use of the image gradients to boost efficiency.

The Hough gradient method has two main stages. On first stage is obtained the binary edge image by applying the Canny edge detector. Additionally, gradients are computed using the first order Sobel derivatives. Then, for every edge points, all points that lies inside a maximum and minimum distance and lie along the gradient are incremented in the accumulator. Those incremented points will be the candidates for possible circle centers.

The algorithm on the second stages selects all center candidates passing a threshold and ensure that no neighbours has higher accumulator value. The selected centers are then sorted by amount of votes in descending order. Beginning at the highest supported center, the algorithm find the radius that best fits the possible edges for this center candidate. The remaining circles are detected in the same way.

Taking this into account, the algorithm provided by OpenCV has four parameters that can be configured: one that define the higher threshold of the two used by Canny edge detector (the lower is half that value); one that define the accumulator threshold for the circle center at the detection stage; and the other two delimit the maximum and minimum radius of the circle.

The problem using this approach was mainly because of the ball used in this work, which even adjusting the parameters referred previously, the results weren't the expected. In indoor tests was possible to detect the ball, but when applied outdoor, it wasn't possible refine those parameters to allow the detection of the ball. This is a consequence

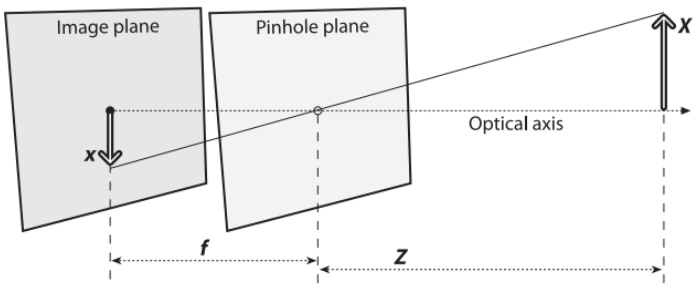


Figure 3.16: Pinhole camera model. [23]

of the lack of consistency of the ball color and the white color parts on the ball, causing a lot of reflection. The conjunction of those two problems leads to a bad edge extraction, influencing the rest of the process. Nevertheless, even if it were possible to detect the ball, the radius of the detected circle presented a considerable variation with the ball static.

Since we are using two cameras, the problem was solved by mounting a stereo system. This was also performed with algorithms provided by OpenCV. This new approach consists of four main steps: realize a stereo calibration with a chessboard; compute the disparity map; realize a 3D reconstruction based on the disparity map and create a point cloud; and in last, apply the method used for 3D data presented on section 3.4 to calculate the coordinates of the center of the ball on the point cloud. The three first steps will be detailed in the next sections.

3.5.1 Camera Model

OpenCV uses the pinhole camera model. This is the simplest, and the ideal, model of camera function. This model is based on modeling the lens of the camera as a pinhole (center of projection) that lets through light rays that intersect a specific point in space and are projected to an image plane. As result, the image on this image plane (also called the projective plane) is inverted and is always in focus, and the size of the image relative to the distant object is given by a single parameter of the camera, its focal length. So the distance from the pinhole aperture to the screen is assumed as the focal length. This is shown in Figure 3.16, where f is the focal length of the camera, Z is the distance from the camera to the object, X is the length of the object, and x is the object's image on the image plane.

To simplify the mathematical relationship between the coordinates of a world 3D point and its projection onto the image plane, the original model suffers a rearrangement to an equivalent form. In this new arrangement, the pinhole and the image plane are swap. A new frontal image plane is generated, where the object is no longer upside down. The point in the pinhole is reinterpreted as the center of projection. The point at the intersection of the image plane and the optical axis is referred to as the principal point. On this new frontal image plane (Figure 3.17), which is the equivalent of the old projective or image plane, the image of the distant object is exactly the same size as it was on the image plane in Figure 3.16. The image is generated by intersecting these rays with the image plane, which happens to be exactly a distance f from the center of projection.

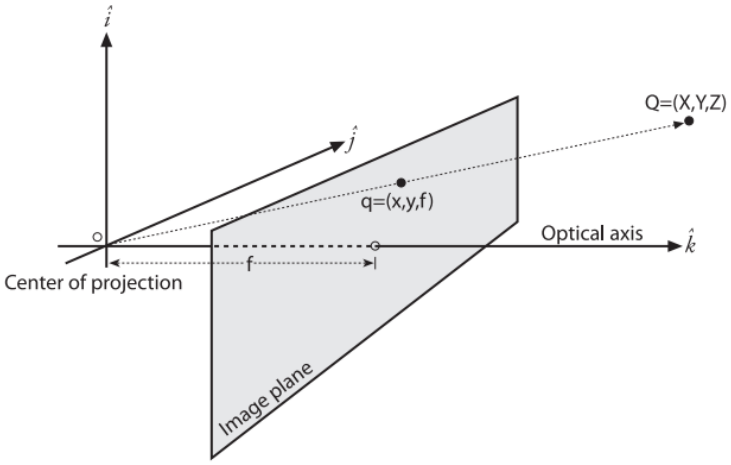


Figure 3.17: Rearrangement of the pinhole camera model. [23]

The intrinsic parameters describes a geometric property of the camera that are used to transform 3D pinhole coordinates of camera frame into 2D image coordinates. The intrinsic matrix of the camera is parameterized as follow:

$$\mathrm{M}=\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The parameters f_x and f_y are focal length, and c_x e c_y represent the principal point. The intrinsic parameters are indifferent of the scenario where the camera are, and once estimated can be used again, since if the focal length does not change.

The relation that transforms the points Q in the physical world with coordinates (X , Y , Z) to the points on the projection screen with coordinates (x , y) is called a projective transform. The projection of the points in the physical world into the camera is proceeded as follow:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

If the world reference frame (X_w, Y_w, Z_w) is different from the camera coordinate frame, another transformation has to be added. This transformation is defined as the extrinsic parameters matrix, which represent the rotation and translation between the world reference and camera frame. The parameters r_{ij} combine rotations along the three main axes and t_i parameters the translation in the three axes. The extrinsic parameters matrix can be defined as:

$$\mathrm{R}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

This way it is possible to establish the relation between the coordinates of the world reference into the camera frame as follows:

$$s \begin{bmatrix} x \\ y \\ w \end{bmatrix} = M \times R \times \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

3.5.2 Stereo Calibration

Stereo calibration is the process of computing the geometric relationship between the two cameras in space. The process to calibrate the two stereo cameras is similar to the process to calibrate just one camera. The main difference, is that in a single camera the calibration returns a list of rotation and translation vectors between the camera and the chessboard views, and the stereo calibration returns a single rotation matrix and translation vector that relate the second camera to the first.

For the calibration of a single camera OpenCV uses a chessboard calibration target, which is used to take several images in different orientations in the camera's fields of view. Then, for each position of the chessboard reference points (i.e. corners) are extracted with sub-pixel resolution from the images and used for estimating the intrinsic matrix of the camera. This method is described in details by Zhang in [24]. An example of the corners extraction can be seen on Figure 3.18.

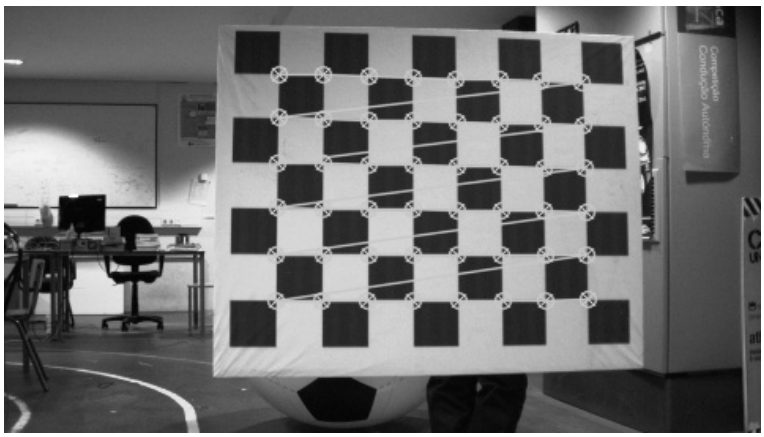


Figure 3.18: Example of the detection of the corners of a chessboard during a calibration.

As referred, the stereo calibration is done almost in the same way as in a single camera calibration, but involves more steps, resulting in the intrinsic parameters of the camera and extrinsic parameters of the stereo rig. This calibration was done with a total of 30 different static positions of the chessboard. Then, in each image the corners of the chessboard were extracted in both images the same way for a single camera, and used as reference points. Next, the reference points of both cameras are used as input on OpenCV to calculate the stereo calibration. In last, the parameters from the calibration are rectified (stereo rectification), which returns the follow parameters:

- R1 - 3 × 3 rectification of rotation matrix for the first camera;
- R2 - 3 × 3 rectification of rotation matrix for the second camera;
- P1 - 3 × 4 projection matrix in the new (rectified) coordinate systems for the first camera;

- P2 - 3 × 4 projection matrix in the new (rectified) coordinate systems for the second camera;
- Q - dispaity-to-depth mapping matrix.

3.5.3 Stereo Imaging

This section describes how the images captured from both cameras can be converted to tridimensional data. Stereo imaging consists of finding the relation of two or more images of the same scenario at different points of view. This principle can be compared with human vision, in which each eye offers a different perspective of the viewed scenario, and give us a tridimensional perception of surrounding environment. So, if we are capable of finding correspondences between images, and knowing the baseline separation between cameras and the intrinsic parameters of stereo system, we can compute the depth information of objects of a certain scenario. To realize the reconstruction from two images to 3D world coordinates, the OpenCV library already offers all the tools necessaries to compute this transformation. This involves four steps:

- Remove radial and tangential lens distortion to obtain undistorted images;
- Adjust for the angles and distances between cameras to obtain rectified image in frontal arrangement (Figure 3.22) a rectification of images. Results in two coplanar images with images rows exactly aligned;
- Find the same features in the left and right camera views. With this correspondences between images, a disparity map is created, where the disparities are the differences in x coordinates on the image planes of the same feature viewed in the left and right cameras;
- Knowing the geometric arrangement of the cameras, reproject the disparity map into depth map by triangulation.

Undistortion and Rectification

The disparity image is easy to compute in the ideal case when two images planes are perfectly aligned. However, in practice that situation is rare with real stereo system, since the two cameras almost never have exactly coplanar, row-aligned imaging planes. The main goal of the rectification is that, reproject the image planes of two cameras so they reside in the exact same plane, with image rows perferctly aligned into a frontal parallel configuration.

The algorithm provided on OpenCV already make the undistortion and rectification of the images, since they are related. OpenCV implements two methods to compute the rectification: Hartley's algorithm [25]; and Bouguet's algorithm. Bouguet's algorithm was the chosen. This algorithm for a given rotation matrix and a translation resulted from the stereo calibration, attempts to minimize the amount of change that a new projection produces for each of the two images in order to reduce the resulting distortion while maximizing the common viewing area. For more information, this algorithm is a completion and simplification of the method first presented by Tsai [26] and Zhang [24; 27]. The algorithm steps represented on Figure 3.19.

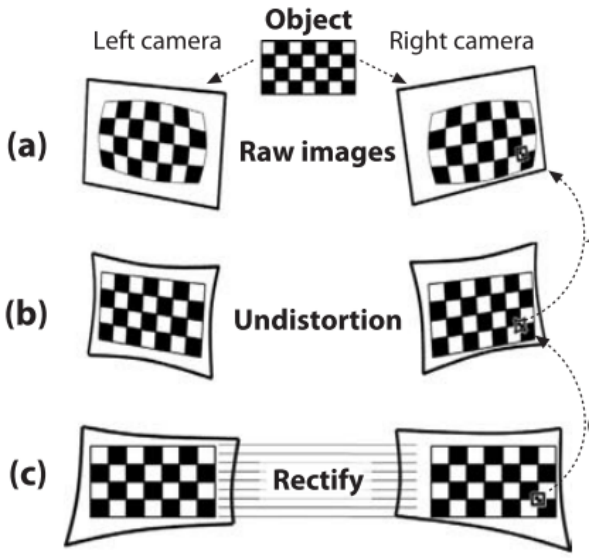


Figure 3.19: Steps of the rectification of an image by OpenCV. [23]

Stereo Correspondences

The stereo correspondences consists of matching 3D points of the same scene in the two different camera views, which can be computed only over the visual areas in which the views of the two cameras overlap. The regions in one image which have no counterparts in the other image, are referred to as occlusions. This is the reason why are obtained better results if the arrangement of the cameras are the best possible near frontal parallel.

In OpenCV are available two different algorithms to calculate correspondences: the block-matching (BM), which is similar to the one developed by Kurt Konolige [28]; and an adaptation of Hirschmuller's semi-global matching (SGM) [29], referred as the semi-global block matching (SGBM). After some tests, was concluded that the SGBM present better results than the BM algorithm, where is possible to observe that the SGBM present a smoother disparity image and with less noise than the BM algorithm. Nevertheless, even being a slower algorithm, and since for this application is not required high frame rates, the process of ball detection is not affected. The images taken by both cameras are present on Figure 3.20, and the resulting disparity image by both algorithms are present on Figure 3.21.

The SGBM method is based on the idea of pixelwise matching of Mutual Information and approximating a global, 2D smoothness constraint by combining many 1D constraints. The algorithm is described in distinct processing steps, assuming a general stereo geometry of two or more images with known epipolar geometry. A detailed description of the algorithm is available in [29].

The implementation of the SGBM on OpenCV have some configurable parameters, in order to obtain the best image disparity depending on the application. Those parameters are the following:

- **minDisparity** - Minimum possible disparity value;
- **numDisparities** - Maximum disparity minus minimum disparity. The value is always greater than zero, and in this implementation must be divisible by 16;

- **SADWindowSize** - SAD (Sum of Absolute Difference) is a measure of similarity between image blocks. This parameter is to define the block size, which must be an odd number ≥ 1;

- **P1** - The first parameter controlling the disparity smoothness;

- **P2** - The second parameter controlling the disparity smoothness. The larger values the values are, the smoother disparity is. P1 is the penalty on the disparity change by plus or minus 1 between neighbor pixels. P2 is the penalty on the disparity change by more than 1 between neighbor pixels. The algorithm requires P2 $>$ P1;

- **Disp12MaxDiff** - Maximum allowed difference (in integer pixel units) in the left-right disparity check. Set it to a non-positive value to disable the check;

- **preFilterCap** - Truncation value for the prefiltered image pixels. The algorithm first computes x-derivative at each pixel and clips its value by [-preFilterCap, preFilterCap] interval. The result values are passed to the Birchfield-Tomasi pixel cost function;

- **uniquenessRatio** - Margin in percentage by which the best (minimum) computed cost function value should "win" the second best value to consider the found match correct;

- **speckleWindowSize** - Maximum size of smooth disparity regions to consider their noise speckles and invalidate. Set it to 0 to disable speckle filtering;

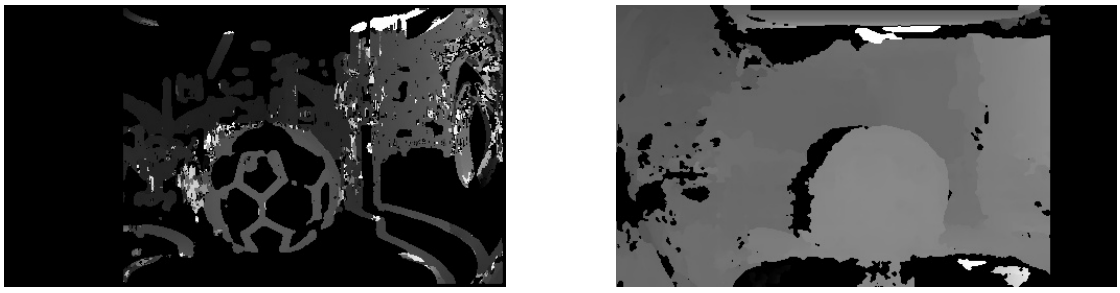
- **speckleRange** - Maximum disparity variation within each connected component. If speckle filtering are activated, the parameter is a positive value, which will be implicitly multiplied by 16;

- **fullDP** - Set it to true to run the full-scale two-pass dynamic programming algorithm. It will consume O(W*H*numDisparities) bytes, which is large for 640x480 stereo and huge for HD-size pictures. By default, it is set to false.

Image camera 1.

Image camera 2.

Figure 3.20: Images of a scenario taken by both cameras.



BM. SGBM.

Figure 3.21: Disparity images of BM and SGBM algorithms.

After some experimental tests, the values that best adjust for this work either for indoor or outdoor are presented on Table 3.1. Also, during those tests was verified that the intensity of light didn't effect much the disparity image on this cameras.

Table 3.1: Parameters settings.

minDisparity	0
numDisparities	192
SADWindowsSize	5
P1	950
P2	2500
Disp12MaxDiff	10
preFilterCap	4
uniquenessRatio	1
speckleWindowSize	150
speckleRange	2
fullDP	false

3D Reconstruction

Knowing the geometric arrangement obtained from the stereo calibration, it is possible reproject the disparity map into depth by triangulation if the geometric arrangement is similar to the one represented on Figure 3.22.

In order to understand the triangulation let's assume that the images are row-aligned and that every pixel row of one camera aligns perfectly with the corresponding row in the other camera. It is also assumed that is possible to find a point P in the physical world in the left and the right image views at p_l and p_r with the respective horizontal coordinates x^l and x^r. This conditions shows that the depth is inversely proportional to the disparity between these views, where the disparity is defined by $d = x^l - x^r$. Figure 3.22 shows that situation, being easily the estimation of the depth Z of the point P by the equation 3.10.

$$\frac{T - (x^l - x^r)}{Z - F} = \frac{T}{Z} \rightarrow Z = \frac{fT}{x^l - x^r} \tag{3.10}$$

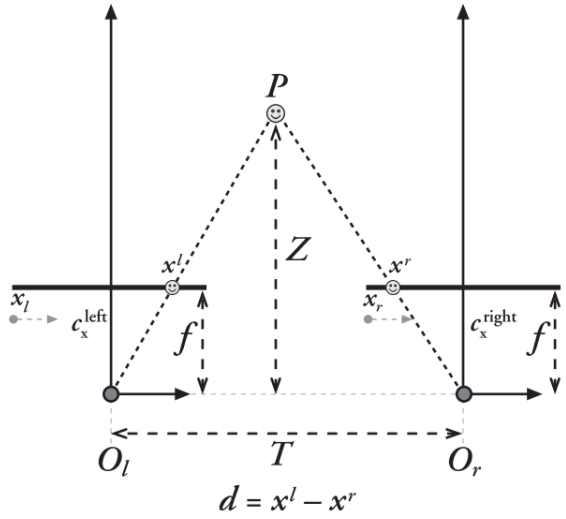


Figure 3.22: Illustration of the triangulation method. [23]

Since depth is inversely proportional to disparity, there is a nonlinear relationship between this two terms. When disparity is near zero, small differences on disparity results in large depth differences. In the other hand, for large disparities, small differences on disparity do not changes the depth by much. As consequence, stereo vision systems have high depth resolution only for objects relatively near the camera. This relation are illustrated on Figure 3.23.

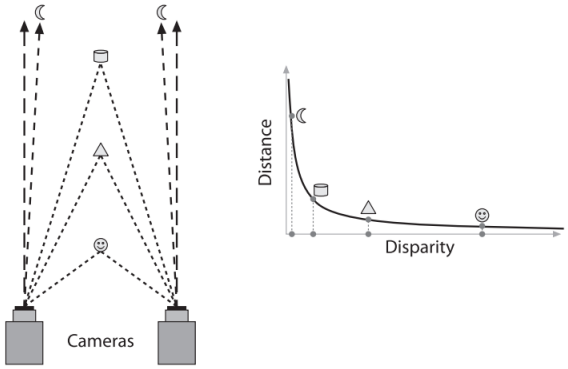


Figure 3.23: Depth and disparity relationship. [23]

Figure 3.24 shows an example of the reprojection of the disparity map into 3D coordinates of our stereo vision system, where the ball is in the field of view of both cameras.

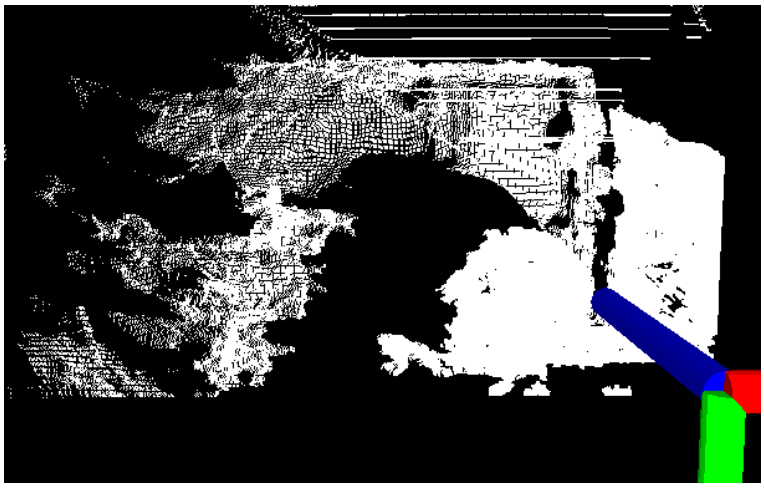


Figure 3.24: Example of an 3D reconstruction of the disparity map.

3D Coordinates of the Ball Center

As referred previously, the center of the ball is calculated as described on subsection 3.4. The resulting 3D points of the reprojection of the disparity map are converted into a point cloud; then, the RANSAC algorithm is used in that point cloud, giving as result the coordinates of the center (x_c, y_c, z_c) of the ball and its radius. Figure 3.25 shows the result of the ball detection applied on the point cloud of the stereo vision system.

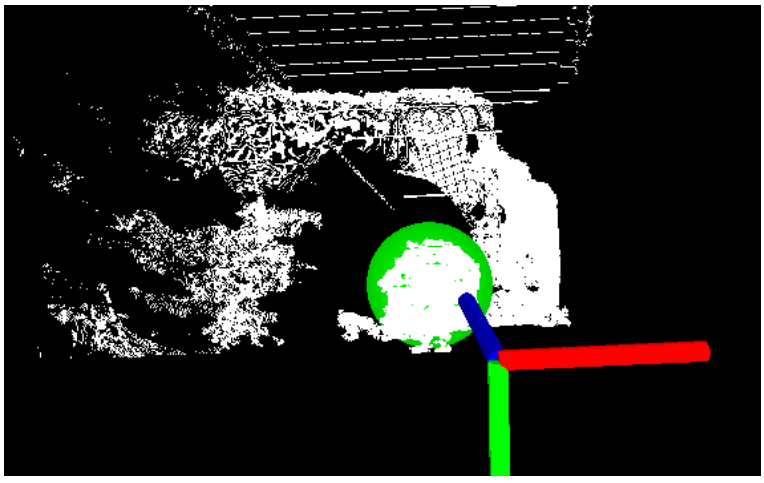


Figure 3.25: Example of the ball detection on 3D point cloud.

3.6 Architecture

As referred previously one of the main advantages of using ROS, is that is possible to have more than one process (node) running at the same time, and exchange information between nodes through a publish/subscribe messaging model into topics. During the process of calibration one of the steps require a synchronized acquisition of points of the center of the ball by all sensors. Thus, each sensor has a node associated to detect the

center of the ball and publish its coordinates into a specific topic. Then, a calibration node has the function of subscribe each topic where the nodes of the sensors are subscribing. Figure 3.26 shows all the processes that run during the calibration and what each of them is publishing and subscribing. Four processes are shown (one for each sensor), but this is scalable for more sensors.

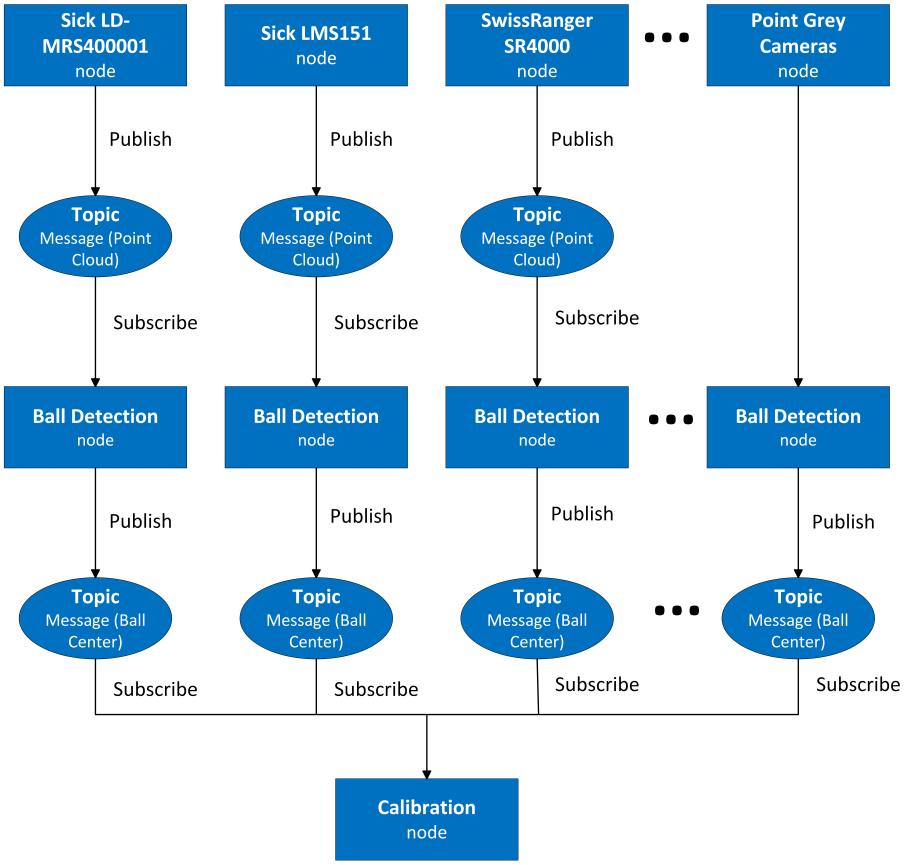


Figure 3.26: Calibration process scheme.

3.7 Point Clouds Acquisition

The last node (calibration node) is responsible for the creation of the point clouds for each sensor, since it is the node that gathers all the information of the other nodes. As known, this point clouds contain the point correspondences between sensors. In order to make that correspondences more accurate possible, some conditions are imposed to consider a valid point during the formation of the point clouds. Considering that there are i sensors and d the distance between consecutive points (the point being evaluated and the last added into the point cloud), the conditions are the following:

- Each sensor have to detect the center of the ball in the same time stamp;
- To prevent the inclusion of more than one point into the point cloud when the ball is static, a minimum distance D_{min} between consecutive points are required. That

means $d_i > D_{min}$;

- Finally to prevent some false detection of the ball in some of the sensors, when the last two condition are fulfilled, there is one last condition. In the perfect scenario, the distance d_i should be equal in all point clouds, although, small differences are implicit due the associated errors in the ball center detection. To prevent that the differences are to large, which could mean a false ball detection, a mean $\bar{d}$ of the distances d_i of each sensor is calculated independently, $\bar{d} = \frac{1}{i}\sum_{j=1}^{j=i} d_j$. Then, the difference Δd_i between the mean $\bar{d}$ and d_i is calculated for all sensors, $\Delta d_i = |\bar{d} - d_i|$. If Δd_i in all sensors is smaller than a certain threshold D_{th}, $\Delta d_i < D_{th}$, the point of each sensor is valid and is added into the respective point cloud. However, when the point clouds are empty this condition cannot be verified.

3.8 Calibration

The 3D transformation estimation purpose is to align the point clouds generated by each sensor. This alignment results on the relative pose (position and orientation) between sensors in a global coordinate frame, such that the overlapping areas between the point clouds match as well as possible. To compute the transformation between sensors an Absolute Orientation algorithm is used. A variant of the Iterative Closest Point (ICP) algorithm is used to estimate the 3D translation and rotation between a pair of point clouds, available on PCL.

The objective is to solve the rigid transformation T that minimizes the error of the point pairs. For that purpose the ICP algorithm provided by PCL has two error minimization metrics: point-to-point and point-to-plane. For this work was applied the point-to-point (Eq. 3.11) metric, where p_n and q_n are 3D points of N pair correspondences from the source and target cloud.

$$E(T) = \sum_{n=1}^{N} ||Tp_n - q_n||^2 \tag{3.11}$$

This metric to minimize the sum of the Euclidean distances between corresponding points corresponds to a least-square problem, which can be solved by using a closed-form using Singular Value Decomposition (SVD) method proposed in [7; 30].

The SVD method uses two sets of corresponding 3D points; one is the source and the other the target. Based on the point correspondences, the cross correlation matrix M between the two centered point clouds can be calculated, after which the eigenvalue decomposition is obtained as follows:

$$M = USV^T \tag{3.12}$$

Then the optical solution to the least-square problem is defined by a rotation matrix R as:

$$R = UV^T \tag{3.13}$$

and the translation from target point cloud (c_t) to source point cloud (c_s) is defined as:

$$t = c_s - Rc_t \tag{3.14}$$

The concept of the ICP algorithm is illustrated on Figure 3.27.

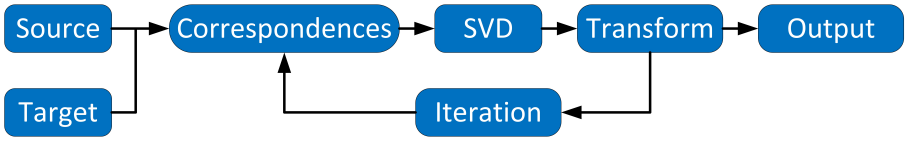


Figure 3.27: ICP overview scheme.

Chapter 4

Results

This chapter describes the experiments that were carried out to test all the functionalities of the new developed calibration method. The tests have special interest in evaluating:

- The consistency in the ball detection in each sensor;
- The consistency in the 3D transformation estimation depending on the number of points used;
- The validation of the method.

4.1 Evaluation of the Consistency in the Ball Detection and the 3D Transformation

The test about the consistency of the 3D transformation does not include the stereo system, because it was not yet integrated on the calibration process at the time. Since the goal is to evaluate the relation of the size of the point clouds on the calibration, the use of the other sensors is already a good indicator of this information. Also, the test of the consistency on the ball detection was not performed for the stereo system since the integration of cameras in this work presented some limitations on the ball detection, being necessary a further study. This limitations will be presented and explained on the evaluation tests.

4.1.1 Ball Detection

In the ball detection test, and in similar conditions, the ball was placed statically at different distances from the sensor and, for each position, 500 samples of the coordinates of the ball center were acquired. From these samples, a mean and the standard deviation were calculated. Figure 4.1 shows the standard deviations of the measurements of the three sensors used on this test.

It is important to refer that the test with the SwissRanger was done only up to 3 m due its small range (maximum of 5 m), because for distances longer than 3 m, the algorithm couldn't detect the ball.

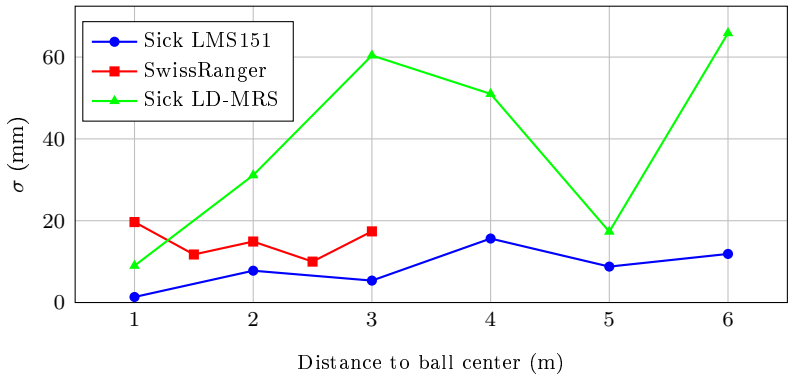


Figure 4.1: Standard deviation of ball center detection for several distances using sets of 500 measurements.

The results show that the variation verified is consistent with the error associated to each sensor (below 10 cm for the Sick LD-MRS, and about 1 cm for the remaining sensors). In the case of the SwissRanger, a greater variation is verified for the closest and furthest distances; it may be due to the proximity of the ball and a lower density of points for further distances, but this is only a suspicions that needs a more detailed study. Nonetheless, the standard deviation in the detection of the ball is not larger than the error associated to the standalone sensors, indicating that the method does not introduce new measurement errors. For the Sick LD-MRS, even with results in accordance with its error, it is difficult to evaluate the interference of the distance of the ball on its detection, which can be possibly explained by the fluctuations in the data provided by the sensor; for example, it is possible to have a set of 500 measurements of a static object at 10 m, where 80% of the measurements have a deviation of ±0.01 m and the remainder 20% have a deviation of ±0.08 m. On the other hand, for the exact same conditions, if a new set of 500 measurements is obtained, a ratio of 50%/50% may be found instead.

4.1.2 Transformation Estimation

In the second test, several calibrations between all the sensors were performed using always the same setup; this was done by using different sizes of the point clouds (number of points), where each point of the cloud corresponds to a different position of the ball during its motion in front of the sensors. Thus, for each size of the point clouds, a set of 20 calibrations was performed having as result the respective matrix of the estimated rigid transformation. The analysis of results compares the translation and rotation; the translation is obtained directly from the matrix of the geometric transformation; considering the rotation matrix (R) from calibration and R_x, R_y and R_z the generic rotations around each axis, R is defined as $R = R_x(Roll)R_y(Pitch)R_z(Yaw)$, which allows to be solved and obtain the Euler angles ($Roll$, $Pitch$, Yaw). Then, as in the first test, a mean and a standard deviation of the translation and Euler angles are calculated, with the difference that the Euler angles are analysed individually. Figure 4.2 shows the results for the translation, and Figure 4.3 presents the results for the Euler angles.

Figure 4.4 shows the setup used for the calibration and Figure 4.5 the estimated positions of the sensors after being calibrated. As expected, the standard deviation decreases with the number of points, and stabilizes at around 20 points; for this test

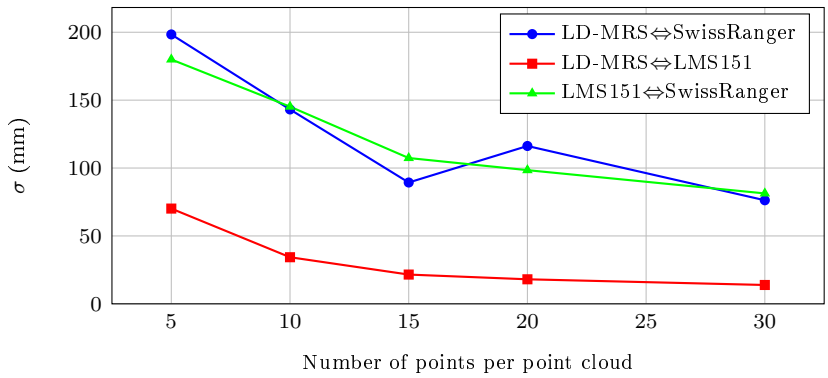


Figure 4.2: Standard deviation of the translation between the three sensor pairs from the 20 calibrations realized for each size of the point clouds.

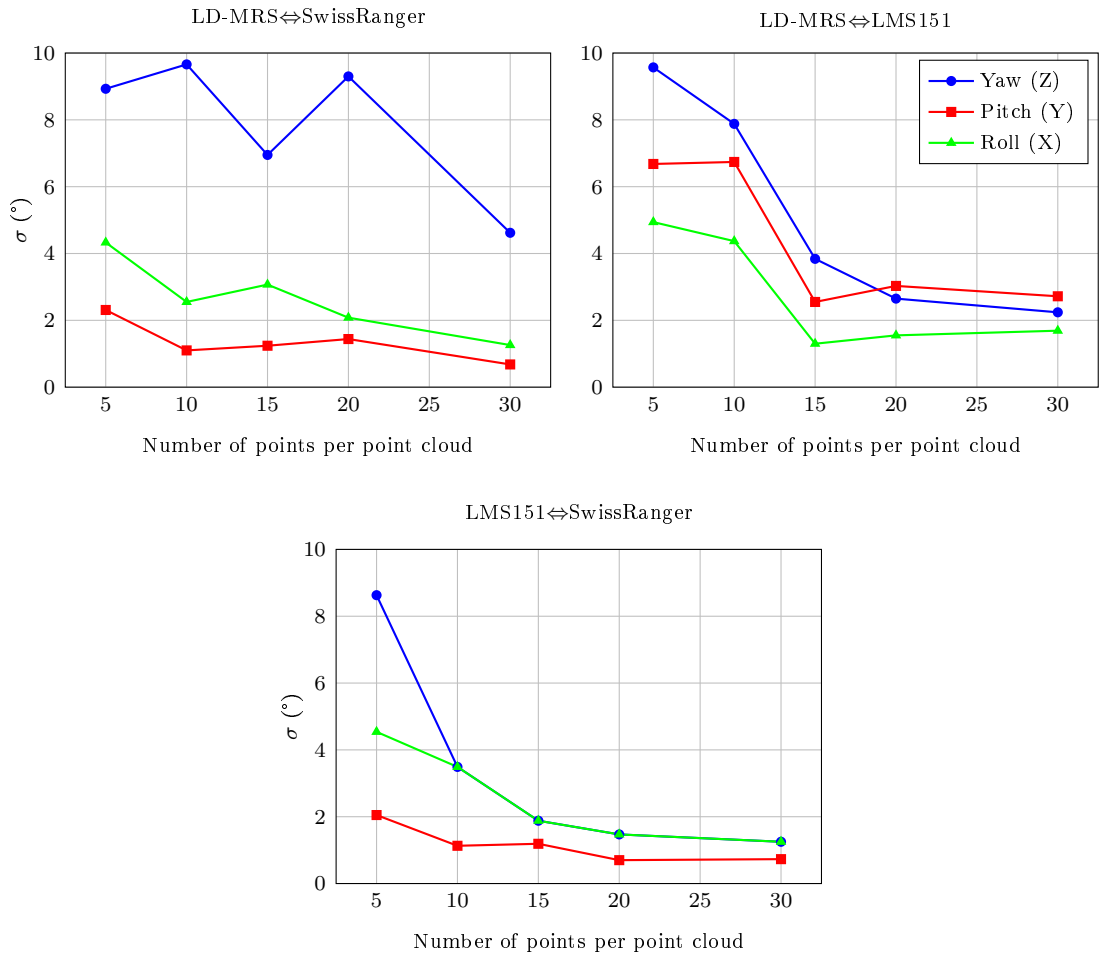


Figure 4.3: Standard deviation of the Euler angles between the three sensor pairs from the 20 calibrations realized for each size of the point clouds.

the minimum distance between consecutive points was 10 cm. The mean variation from point clouds of 20 points is lower than 10 cm and about 4°or 5°, which, once again, is in the range of the standalone sensors.

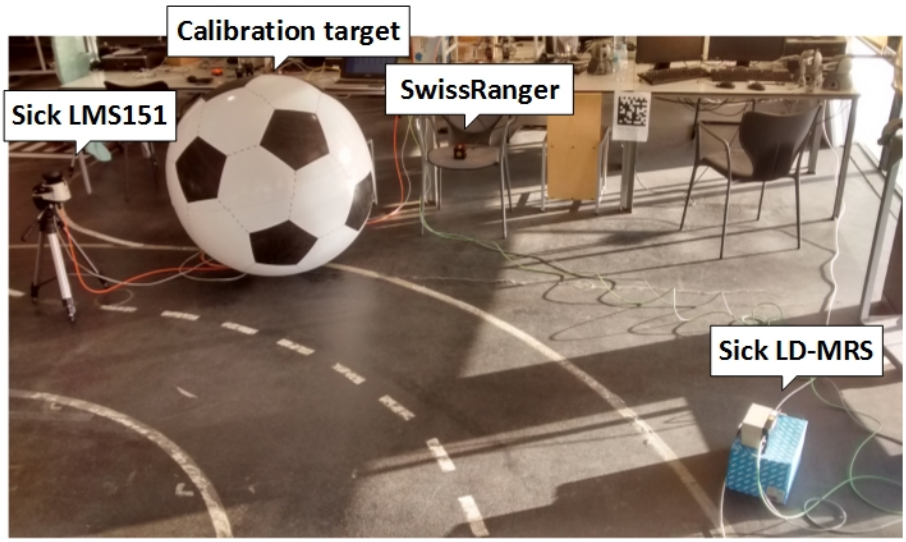


Figure 4.4: Layout of the sensors on the scene.

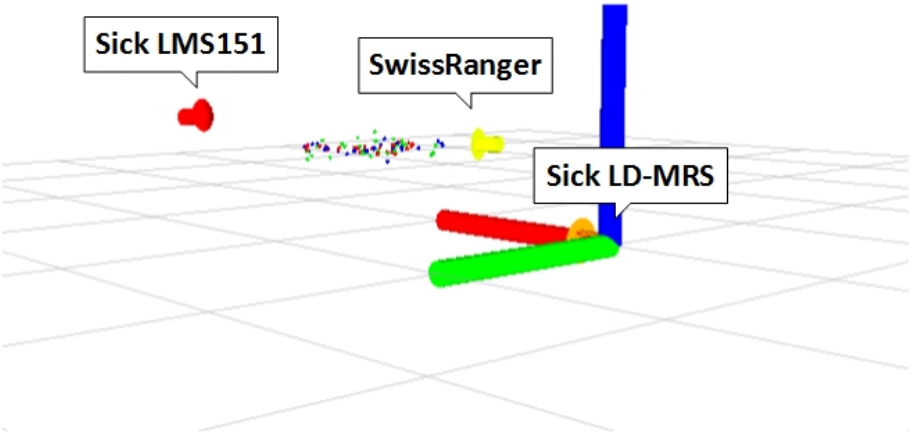


Figure 4.5: Positions of the sensors after the calibration.

4.2 Validation of the Method

The difficulty of having a ground-truth makes complicated the validation of the method, which would allow the comparison of the estimated transformation with the exact transformation. It is very difficult to guarantee with precision if the calibration obtained is correct, given the difficulty to measure exactly the correct pose between pair of sensors. The problem of evaluating the pose between sensors is particularly difficult with rotations (it is possible to have a fairly reasonable evaluation of the translation with simple

measures), which is critical since small errors in rotation may result in large error in the final registered data.

Having in account how difficult is to prove the validity of the method, the best solution found was to perform a calibration of all the devices with respect to the laser with better accuracy. Then, the point cloud of the reference sensor is used as ground-truth, and on the remainder is applied the resulting transformation from the calibration to evaluate how well the point clouds fit in the reference by calculating the absolute mean error and the standard deviation of the corresponding points relatively to the ground-truth.

This test was performed in a vehicle (AtlasCar) to evaluate the method on real conditions. And since this tests are outdoor, the devices used are the ones that will have some functionality on the AtlasCar project, which means that the set of sensors does not include the SwissRanger. The main reason to exclude the SwissRanger is that the sensor does not work outside being affected by external light that interferes with the infrared light projected by the system.

The sensors used in this experiment are: the Sick LD-MRS, the two Sick LMS151 and the stereo system. One of the Sick LMS151 was the chosen sensor to be the reference and the remainder are calibrated relatively to it. The experience has the following steps:

- Chose one of the sensors as reference (in this case a Sick LMS151 was chosen given this is the sensor with better accuracy);
- Perform the point cloud acquisition for each sensor and calibrate them with respect to the reference frame;
- Apply the resulting transformation on the point clouds of the calibrated sensors;
- Calculate the error of each point belonging to the point cloud of the calibrated sensors relatively to the corresponding points of the ground-truth (point cloud of the reference sensor);
- Calculate the mean error and standard deviation of the of the previous step.

The acquisition of the point clouds were performed with two different approaches: i) following a pattern, by forming a grid as illustrated on Figure 4.6 (it is just an example of the intended); ii) acquire random points during the motion of the ball in front of the sensors. The idea of acquiring a pattern is to perform a study about the influence of the distribution of the points on the calibration by selecting different points from the pattern. The acquisition of random points is to evaluate the calibration method in a normal situation. Figure 4.7 shows the setup mounted on AtlasCar, where can be seen that the Sick LD-MRS is placed on the top of a box because there is not yet a support to apply it in the car, nonetheless, its location will not be much different from where it was placed.

4.2.1 Calibration with a Pattern

To perform the acquisition with the ball moving along a grid/pattern, several markers were placed in the floor and the ball was placed as close as possible to each marked position, forming a point cloud for each sensor. Figure 4.10 shows the markers placed on the floor and the resulting point cloud on the Sick LMS151 sensor.

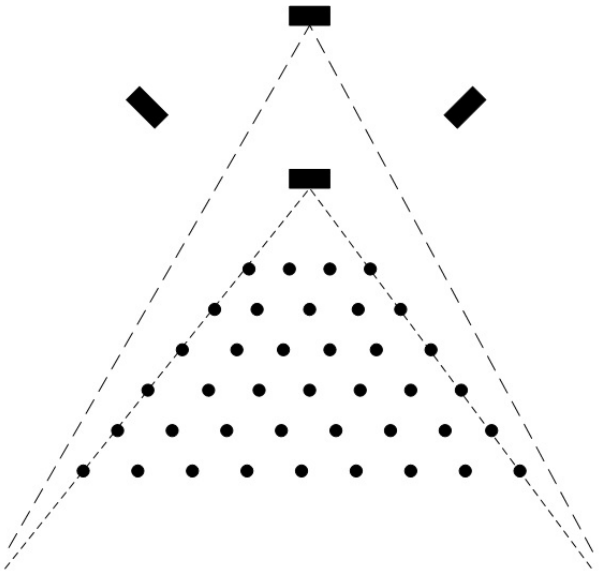


Figure 4.6: Illustration of the pattern to be acquired forming a grid. The sensors are represented by the black rectangles, and the points are the placement of the target (sphere).

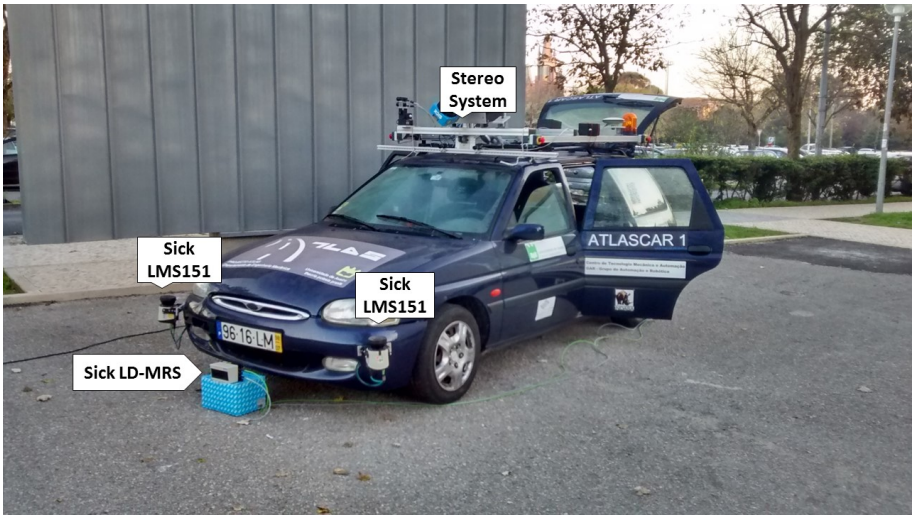


Figure 4.7: Setup used on the validation test and the sensor positions.

Floor marks. Resulting point cloud.

Figure 4.8: Markers placed on the floor and a resulting point cloud on the Sick LMS151 sensor.

The resulting point clouds of each sensor were divided in two sub sets of points. One with the central points of the point cloud, which will be called inner points, and the second with the remaining points, which will be called outer points. This two different sub-sets were used to evaluate the influence of the ball position (namely the size of the area used for calibration). Thus, evaluate the influence of the area size on the transformation estimation. Figure 4.9 shows the selected inner points in red and the outer points in green.

Figure 4.10 shows the point clouds of each sensor with respect to their own frame. On the left the point clouds with the inner points and on the right the point clouds with the outer points.

The calibration of the sensors with respect to the reference on both cases (with the inner and outer points) is performed. Then, the resulting transformation was applied on the point cloud of each sensor. The result for both cases is illustrated on Figure 4.11; on left is the result for the calibration with the inner points, and on right the result using the outer points. This figure shows, for each pair of calibrated sensors, the fitting of the point clouds of the calibrated sensor on the point cloud of the reference sensor (used as a sort of ground-truth). There are some points between LIDAR lasers that do not seem to have a fitting, which happens when two points are overlapped.

To evaluate the quality of the calibration, the distance between each corresponding point of the calibrated sensors to the reference sensor was calculated. The measures obtained were the standard deviation and the error associated to each corresponding point. Then, the absolute mean error is calculated for each calibrated sensor. The resulting standard deviation and the absolute mean error of the calibration using the inner points are shown on Table 4.1, and using the outer points on Table 4.2.

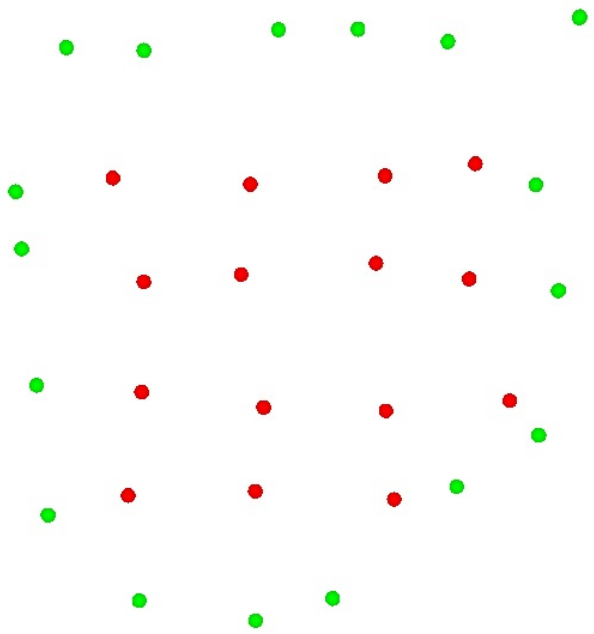


Figure 4.9: The two sub sets of points from a point cloud: inner points in red and outer points in green.

Table 4.1: Resulting standard deviation and absolute mean error for each calibrated sensor using the inner points.

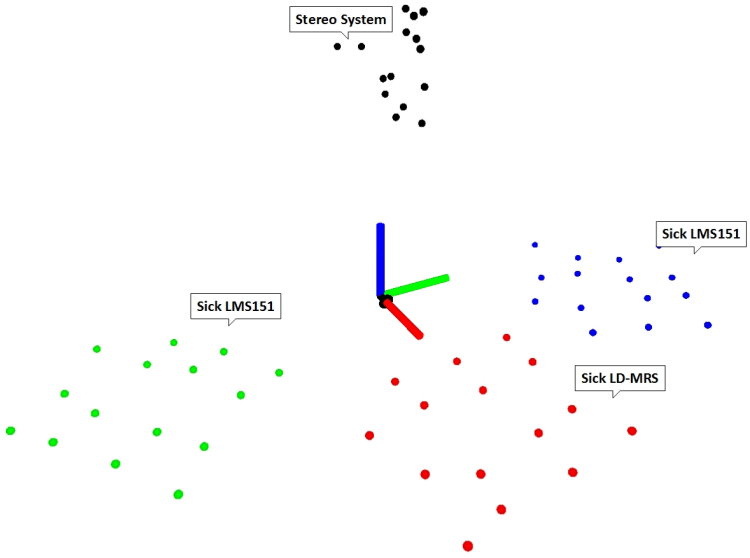
	Standard deviation [cm]	Absolute mean error [cm]
Sick LMS151 - Sick LMS151	2.897	2.292
Sick LMS151 - Sick LD-MRS	3.758	3.242
Sick LMS151 - Stereo system	38.470	31.335

Table 4.2: Resulting standard deviation and absolute mean error for each calibrated sensor using the outer points.

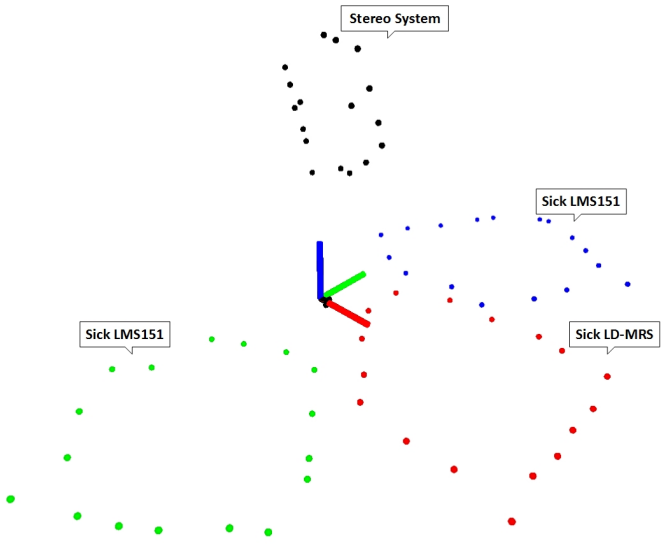
	Standard deviation [cm]	Absolute mean error [cm]
Sick LMS151 - Sick LMS151	3.803	2.618
Sick LMS151 - Sick LD-MRS	4.544	3.729
Sick LMS151 - Stereo system	76.094	63.568

In order to have a better idea of the quality of the calibration, the sensors were placed on a CAD model of the AtlasCar. The result is shown on Figure 4.12, where the sensors are represented by arrows pointing in the X axis direction of their own referential system.

The pattern acquired with the lasers is similar to the expected, the differences that are noticed it is because of a bad placement of the ball on the marks, and the nearby objects to the markers. However, on Figure 4.10 can be observed that the result with the stereo system was poor. This can be a cause of several factors, from a bad calibration of the stereo system to the scene where the pattern was acquired, which has a lot of

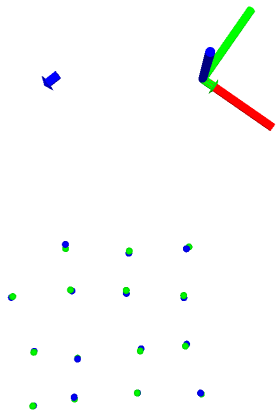


Inner points.

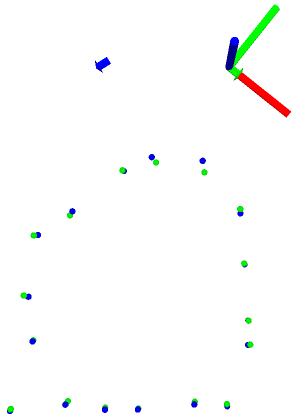


Outer points.

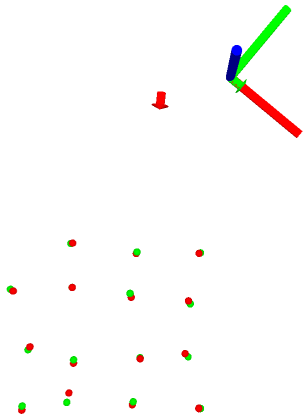
Figure 4.10: Representation of the sensors point clouds not yet calibrated. On the left the inner points and on the right the outer points.



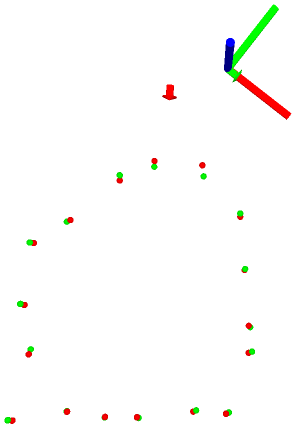
Sick LMS151 - Sick LMS151.



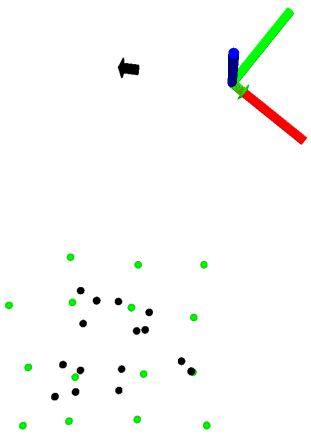
Sick LMS151 - Sick LMS151.



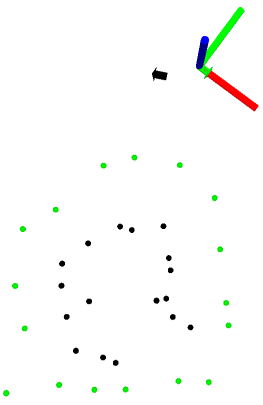
Sick LMS151 - Sick LD-MRS.



Sick LMS151 - Sick LD-MRS.

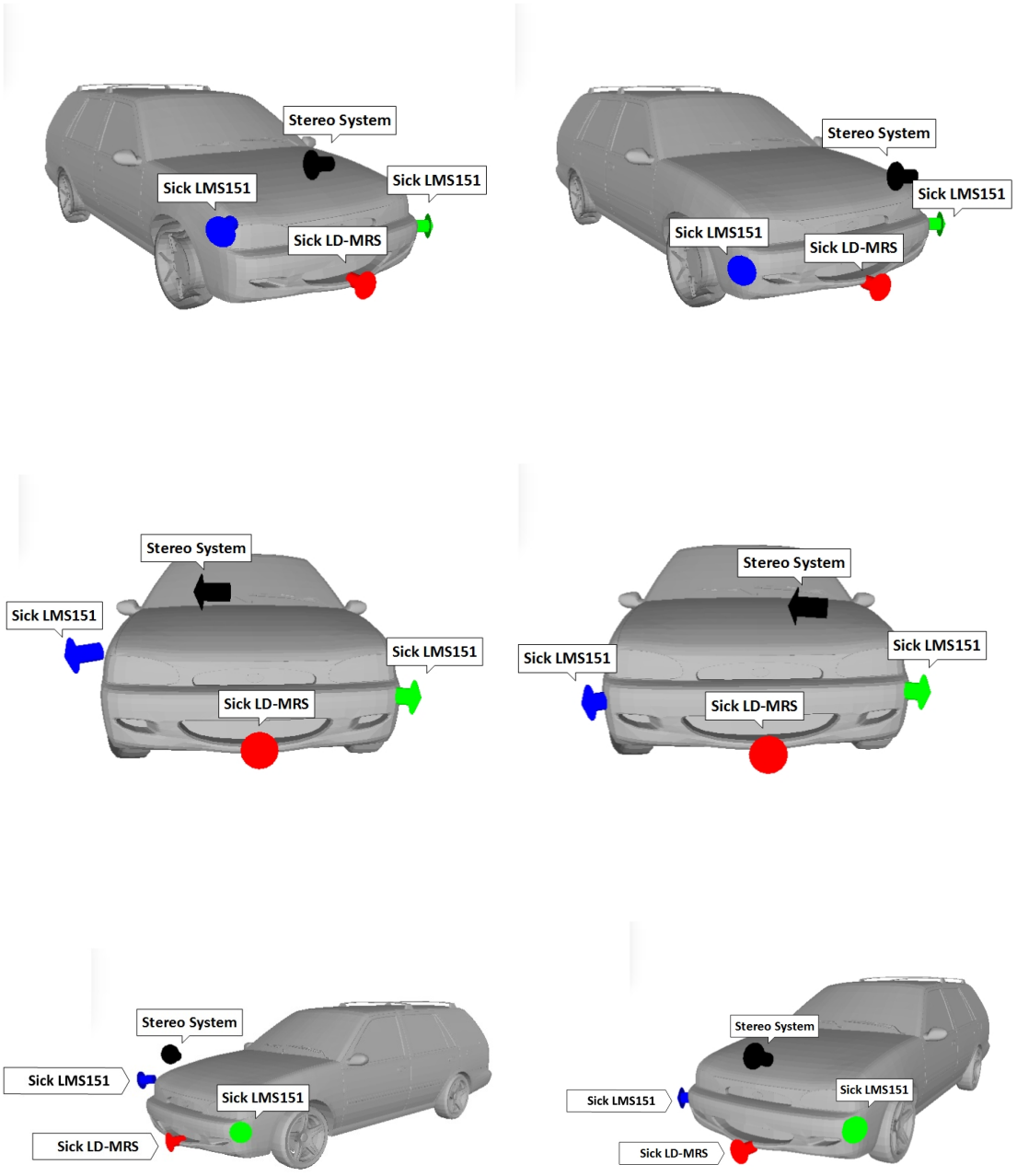


Sick LMS151 - Stereo System.



Sick LMS151 - Stereo System.

Figure 4.11: Fitting of each pair of calibrated sensors. The figures on the left are from the calibration with the inner points, and on the right with the outer points.



Calibration using the inner points. Calibration using the outer points.

Figure 4.12: Representation of the sensors positions in a CAD model of the AtlasCar on both calibrations.

close objects that may interfere on the ball detection. Also, due to the reflection of light on the ball and its large areas with uniform color, the disparity map shows some noise that affects the re-projection of the ball to 3D coordinates. The re-projection of the ball instead of being a perfect spherical surface, presents an irregular spherical surface, and sometimes is hard to detect as being part of the ball.

As consequence of the bad pattern acquisition by the stereo system, its calibration presents a big standard deviation and absolute mean error for both calibrations as presented on Table 4.1 and 4.2.

Analysing the results for the lasers calibration on Table 4.1 and 4.2, the results between both calibrations does not differ much. The standard deviation and the absolute mean error values are slightly bigger on the outer points, which is related with the furthest measures of the ball center, however, the difference is so small that it becomes irrelevant. However, this does not mean a better calibration, only that the point clouds fit better. Looking at Figure 4.12, and comparing with the setup used (Figure 4.7), the displacement of the lasers on the car model seems to be better on the calibration with the outer points. This result was expected, since the number of possible positions for the lasers to fit two point clouds are bigger if the points cover a small area.

There are three conclusions from this experiment:

- The stereo system needs to be improved and the ball texture should be replaced by a different one, or an alternative implementation should be used for the ball detection;
- The fitting of the point clouds are not much affected by the size of the covered area that contains the corresponding points;
- The use of a larger covered area seems to improve the accuracy of the calibration.

4.2.2 Calibration with Random Points

The calibration method does not need to follow any particular pattern with the corresponding points. Thus, this test is to evaluate a calibration in a normal use by acquiring points in random positions. Differently from the previous test, this one was realized in an open area without close objects that may interfere on the ball detection. Based on the results obtained in the previous tests about the influence of the number of points on the transformation estimation, we chose a number of 25 corresponding points per point cloud. The resulting point clouds of the calibration process for each sensor with respect to their own frame are shown on Figure 4.13.

Again, the calibration of the sensors was computed with respect to the one of them taken as reference (the Sick LMS151). Then, the transformation estimation was applied on the corresponding point cloud. The fitting of the point clouds on the ground-truth is represented on Figure 4.14 in different perspective by pairs of calibrated sensors.

As in the previous test, the position of the sensors after calibration are placed on the car model. The arrangement is shown on Figure 4.15 along with the real setup for comparison.

The point cloud in Figure 4.14 fits better on the calibration of the Sick LMS151 and the Sick LD-MRS, meaning a smaller error on the calibration. On the other hand, the stereo system presents a larger disparity, having as consequence a calibration with a larger error.

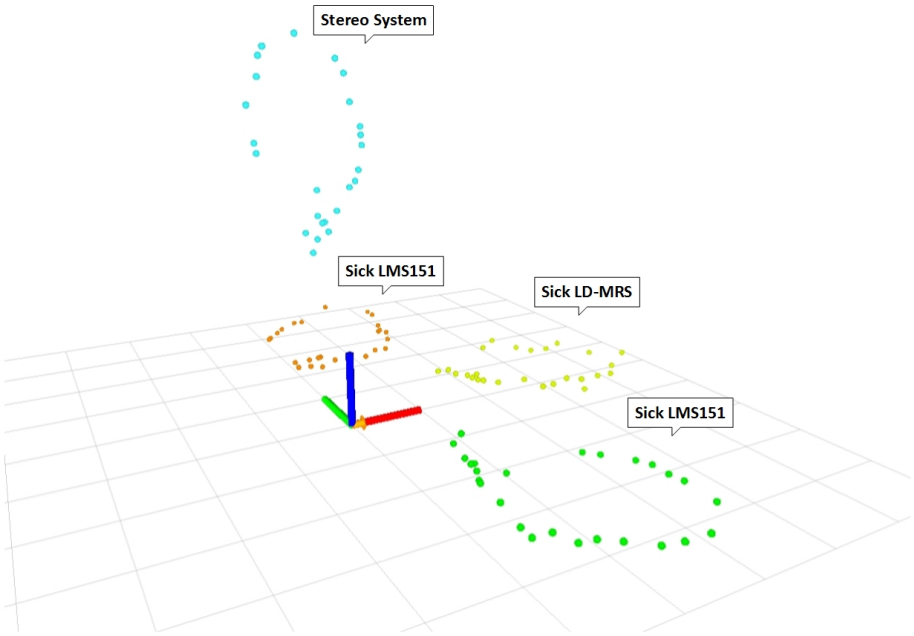


Figure 4.13: Resulting point clouds for each sensor (yet uncalibrated).

Figure 4.15 shows that the obtained calibration and the real setup appear to be fairly consistent for the lasers. Although the same it is not verified for stereo system; the problem appear to be in the 3D reconstruction from the disparity map since the whole 3D cloud of points seems to be to close to the car. This might indicate problems in the stereo rig calibration.

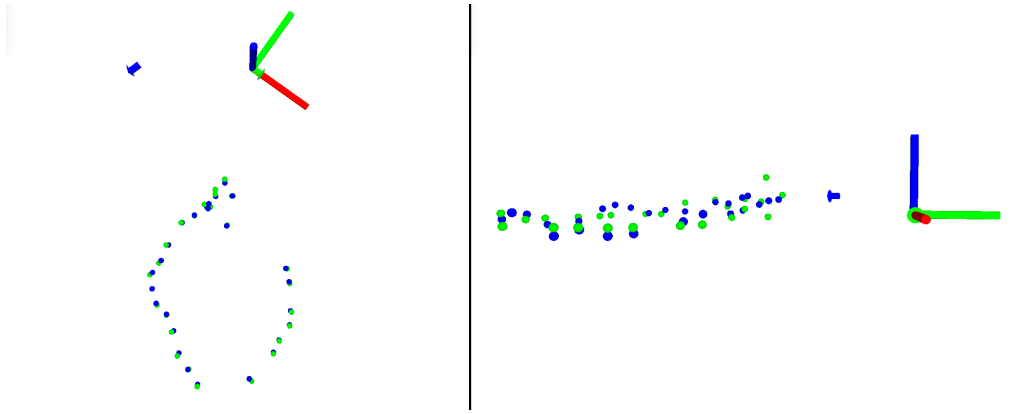
The evaluation of the calibration was performed exactly as in the previous test. The standard deviation and the absolute mean error for each calibrated sensor is presented on Table 4.3.

Table 4.3: Calculated standard deviation and absolute mean error of the calibration.

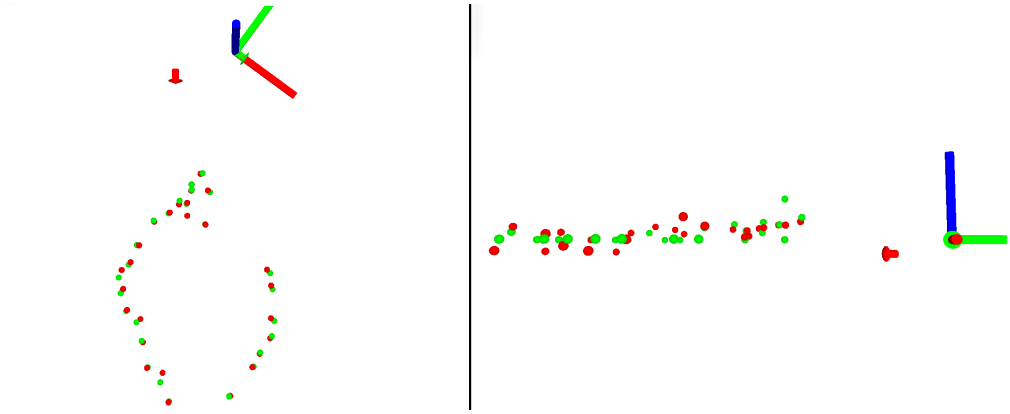
	standard deviation [cm]	absolute mean error [cm]
Sick LMS151 - Sick LMS151	3.114	2.367
Sick LMS151 - Sick LD-MRS	5.994	4.157
Sick LMS151 - Stereo system	21.488	18.189

The results on Table 4.3 for the standard deviation and the absolute mean error between lasers is similar to the ones obtained on the previous test. However, the calibration of the stereo system improved significantly. This may be related with the change of scenery where the calibration was performed, since in this scenario close objects that may interfere on the ball detection did not exist. The error is still too large to allow a real use of the stereo system and further investigation on the topic is necessary to better understand the causes of this large error and improve the process.

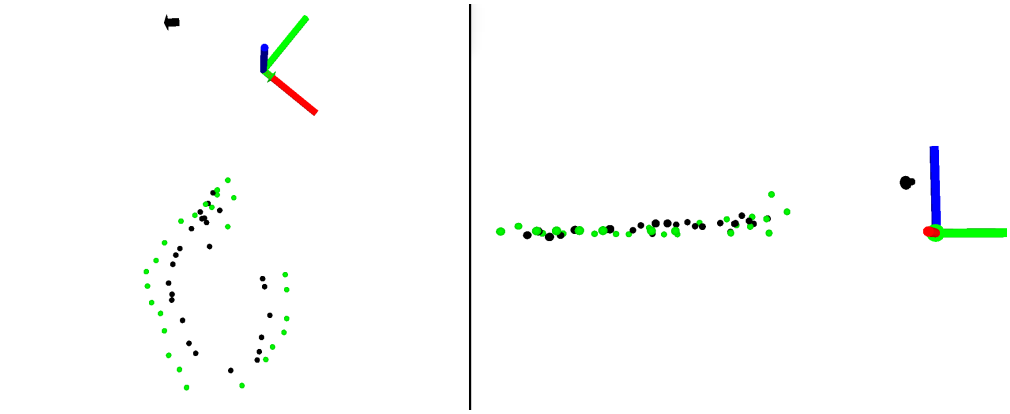
The small error and standard deviation associated between the two lasers Sick LMS151 was expected, since they are the devices with smaller error ($\pm 1cm$), however, the result



Sick LMS151 - Sick LMS151.

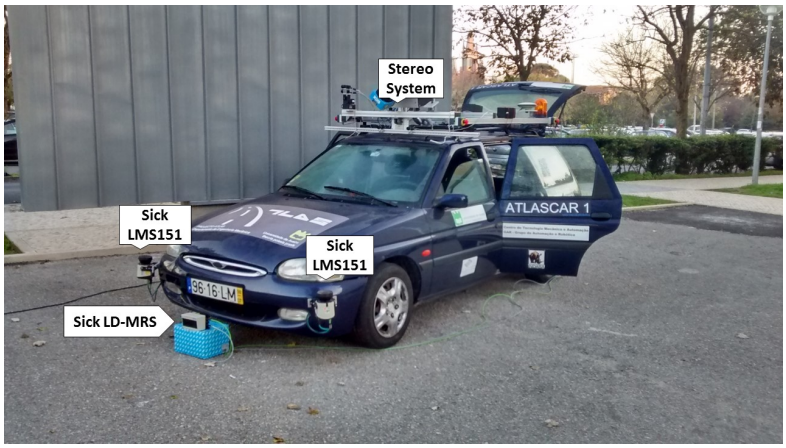


Sick LMS151 - Sick LD-MRS.

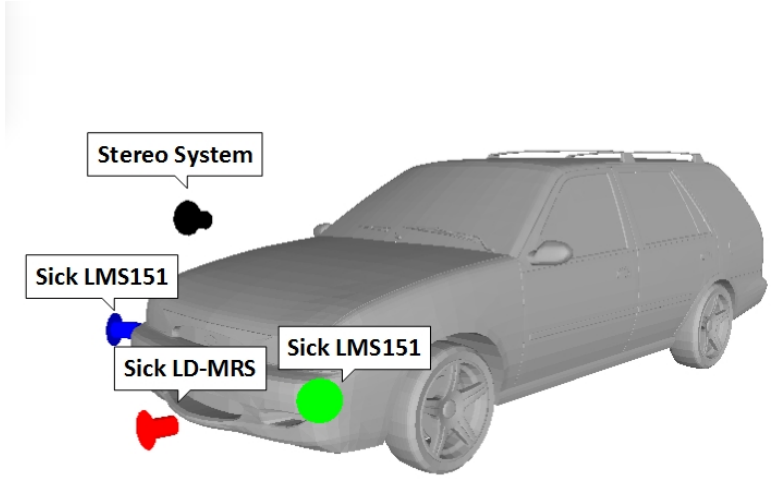


Sick LMS151 - Stereo System.

Figure 4.14: Fitting of the point clouds of each calibrated sensor into the ground-truth from two perspectives.



Sensors setup on the AtlasCar.



Sensors positions on the car model after the calibration.

Figure 4.15: Displacement of the sensors on the AtlasCar and the car model after the calibration for comparison.

between the laser Sick LMS151 and the Sick LD-MRS was surprising, since the calibration error is smaller than the error associated to the Sick LD-MRS ($\pm 10cm$). This results prove that the method for the ball detection on the Sick laser does not provide more error than its associated intrinsic error. Also, since the calibration algorithm uses several points and minimizes the sum of the point pairs error, a point with greater error does not affect much the calibration result.

The largest error on the stereo, once again, may be related to reasons pointed previously, such as the reflection of light on the ball, and its big areas with uniform color.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

A new automatic calibration of multiple LIDAR sensors and cameras was developed and successfully implemented. The algorithm uses a ball as calibration target, allowing the detection of its center by all sensors along successive positions. The only initial parameters required concerns the position of the LIDAR sensors relatively to the ball (whether it is above or below the equator of the ball).

To perform the detection of the ball and posteriorly the calibration, the algorithm uses the LIDAR sensors clustering techniques, arc properties and trigonometric relationships to estimate the center of the ball; in the case of the SwissRanger, the data is processed using a PCL consensus algorithm for the ball center detection; the two cameras are used to build a stereo system, which by a combination of algorithms of OpenCV and PCL, allows the ball center to be detected. The stereo calibration, the calculation of the disparity map and 3D reconstruction are performed using OpenCV, and the detection of the ball with a PCL consensus algorithm. All the different approaches applied to the various types of sensors for the ball detection proved to be suitable for the function, and did not increase the error that each sensor already has associated. However, even the process for the cameras being suitable, some improvements are required.

The calibration estimation between sensors was done based on correspondences. The ball was placed in motion allowing its detection by all sensors for the creation of a point cloud for each sensor. The number of corresponding points influences the calibration quality. 15 seems to be a reasonable threshold according to some experiments performed.

Other objective of this method was also proven, which is related to its capacity to support various types of sensors. That was possible because of the variety of sensors used on this work, which forced this method to deal with different types of data (2D, n×2D, 3D and image). As a matter of fact, this method can handle various types of sensors, as long as it is possible to detect the center of the ball in each of them.

In summary, the good results on the consistency tests for the ball detection and transformation estimation were confirmed with the validation tests by the small errors that were obtained on the calibration of the LIDAR lasers. However, the stereo system presented some limitations on the ball detection, having consequences on the accuracy of its calibration.

5.2 Future Work

For future work, a detailed study about the calibration of cameras should be carried out. The process of the cameras calibration is a good start, however, they presented the worst results. An improvement of the method used for the ball center detection is necessary, or a development of an alternative method for the calibration of individual cameras. Thus, if so, it is advisable to use of a ball with a uniform color. Having a robust process for the ball detection, the calibration of cameras might be as good as the calibration of the LIDAR lasers done in this work.

It would be interesting to create some validation tools to automatically evaluate the quality of the resulting calibration, in order to avoid a potential bad calibration; that could be done by doing something similar to the test performed to prove the validation of the process. That means, after calibrating the sensors, apply the transformations on the respective point clouds and calculate the mean error of the points, exactly as it was done on the validation test. If the mean error in some calibrated sensor is larger than a certain threshold, another calibration should be done.

To remove the necessity of a previous information related with the position of the LIDAR sensor relatively to the ball, a priori process should be implemented relatively to the calibration method. Put the ball bouncing around in front of the sensors, and study the diameter variation of the ball sections, might be a good starting point. With that this method could support any kind of configuration easily.

In order to simplify the use of the method, an interface could be developed to be a more user friendly method; so, it could be used by everyone, without the necessity of using line commands.

Finally, more directly related to the AtlasCar, a support for the application of the Sick LD-MRS shall be done, and applied in the car in the more adequate position for its function. Thus, after calibrate all the sensors on board the car, gather all the information with respect to a reference frame and define the navigable areas for the car.

References

[1] Santos, V., Almeida, J., Avila, E., Gameiro, D., Oliveira, M., Pascoal, R., Sabino,R., Stein, P.: ATLASCAR - Technologies for a computer assisted driving system on board a common automobile. In: 13th International IEEE Conference on Intelligent Transportation Systems (ITSC2010). Madeira Island, Portugal. (2010)

[2] Y. Bok, D. Choi and I. Kweon: Generalized laser three-point algorithm for motion estimation of camera-laser fusion system. In IEEE International Conference on Robotics and Automation (ICRA), pp. 2880-2887, 2013.

[3] J. Shi and C. Tomasi: Good Features to Track. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994, pp. 593-600.

[4] D. Nister: A Minimal Solution to the Generalised 3-Point Pose Problem. In Proceedings of the IEEE Computer Society Coriference on Computer Vision and Pattern Recognition, 2004, pp. 560-567.

[5] Y. Bok, D.-G. Choi, Y. Jeong and I. S. Kweon: Capturing Citylevel Scenes with a Synchronized Camera-Laser Fusion Sensor: In Proceedings of the IEEE/RSf International Conference on Intelligent Robots and Systems, 2011, pp. 4436-4441.

[6] S. A. Rodriguez F., V. Fremont and P. Bonnifait: Extrinsic calibration between a multi-layer lidar and a camera. In IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 214-219, 2008.

[7] K. S. Arun, T. S. Huang, and S. D. Blostein: Leastsquares Fitting of two 3-D point sets. In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol.9, no. 5, pp. 698-700, 1987

[8] G. Chao and J. Spletzer: On-Line Calibration of Multiple LIDARs on a Mobile Vehicle Platform. In IEEE International Conference on Robotics and Automation (ICRA), pp. 279-284, 2010.

[9] M. Lobo, L. Vandenberghe, S. Boyd and H. Lebret: Applications of second-order cone programming. In Linear Algebra and its Applications, Special Issue on Linear Algebra in Control, Signals and Image Processing, vol. 284, pp. 193-228, 1998.

[10] S. Debattisti, L. Mazzei, M. Panciroli: Automated Extrinsic Laser and Camera Inter-Calibration Using Triangular Targets. In IEEE Intelligent Vehicles Symposium (IV), pp. 696-701, 2013.

[11] E. Fernandez-Moral, J. Gonzalez-Jiménez and V. Arévalo: Extrinsic calibration of 2D laser rangefnders from perpendicular plane observations. In The International Journal of Robotics Research, 2015.

[12] Almeida, M., Dias, P., Oliveira, M., Santos, V.: 3D-2D Laser Range Finder calibration using a conic based geometry shape. In Image Analysis and Recognition. Ed. by Aurélio Campilho and Mohamed Kamel. Springer Verlag, pp. 312-319, 2012.

[13] Besl, P.J. and H.D. McKay. A method for registration of 3-D shapes. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14(2), pp. 239-256, 1992.

[14] W. Schroeder, K. Martin , B. Lorensen. The Visualization Toolkit - an object oriented approach to 3D graphics. 4th ed. Kitware, 2006.

[15] Open Source Computer Vision Library (OpenCV), 2015. [Online]. Available: http://opencv.org/. [Accessed 21/10/2015].

[16] R.B. Rusu and S. Cousins: 3D is here: Point Cloud Library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), pp. 1-4, 2011.

[17] M.Quigley, E. Berger and A. Y. Ng.: STAIR: Hardware and Software Architecture. In AAAI 2007 Robotics Workshop, 2007.

[18] M. Pereira, V. Santos and P. Dias: Automatic calibration of multiple LIDAR sensors using a moving sphere as target. In Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, vol. 1, pp. 477-486, 2015.

[19] Coimbra, D.: "LIDAR Target Detection and Segmentation in Road Environment", Master's thesis, Universidade de Aveiro (2013)

[20] J. Xavier, M. Pacheco, D. Castro and A. Ruano: 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, 2005.

[21] M. A. Fischler and R. C. Bolles: "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", Communications of the ACM, Vol 24, pp 381-395, 1981.

[22] C. Kimme, D. Ballard and J. Sklansky: Finding circles by an array of accumulators. In Communications of the ACM, vol. 18(2), pp. 120-122, 1975.

[23] G. Bradski and A. Kaehler: Learning OpenCV: Computer Vision with the OpenCV Library. In O'Reilly Media, 2008.

[24] Z.Zhang: A flexible new technique for camera calibration. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 22, pp. 1330-1334, 2000.

[25] R. I. Hartley: Theory and practice of projective rectification. In International Journal of Computer Vision, vol 35, pp. 115?127, 1998.

[26] R.Y. Tsai: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-self TV cameras and lenses. In IEEE Journal of Robotics and Automation, vol 3, pp. 323-344, 1987.

[27] Z.Zhang, P.-S. Tsai, J.E. Cryer and M. Shah: Flexible camera calibration by viewing a plane from unknown orientations. In IEEE International Conference on Computer Vision, vol. 1, pp. 666-673, 1999.

[28] K. Konolige: Small vision system: Hardware and implementation. In Proceedings of the International Symposium on Robotics Research, pp. 111-116, 1997.

[29] H. Hirschmuller: Stereo Processing by Semiglobal Matching and Mutual Information. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 30, pp. 328-341, 2008.

[30] B. K. P. Horn: Closed-form solution of absolute orientation using unit quaternions. In Journal of the Optical Society of America A, vol. 4, no. 4, pp. 629-642, 1987.