POLITECNICO DI TORINO

M.Sc. in Mechatronic Engineering

Master Thesis

Study and development of calibration algorithms of multicamera systems for precision agriculture robotics

Supervisors
Prof. Alessandro Rizzo
Ing. Antonio Petitti

Candidate
Mariangela Autera

A.Y. 2020-2021

Summary

This thesis has been developed in collaboration with STIIMA-CNR (Sistemi e Tecnologie Industriali Intelligenti per il Manufaturiero Avanzato - Consiglio Nazionale delle Ricerche) located in Bari.
The aim of this thesis is to propose new techniques in the areas of mobile robotics. In particular, we studied innovative data processing techniques devoted to the estimation of the relative pose of a multi-camera system, which can be used also on mobile robots. The estimated pose is obtained thanks to heterogeneous sensors mounted on the platform. So, a method of extrinsic calibration is used applied to a multi-camera system.
Later, the already calibrated multi-camera systems will be used for both robot localization and for the analysis of the surrounding environment.
In order to validate all the methods, we utilize algorithms applied in general for multi-camera systems, but we have validated them with a minimum number of cameras, i.e. two. These cameras are rigidly coupled each other in space. Each camera captures, in time, a cloud of 3D points and estimates the trajectory traveled in space, through SLAM algorithms. Based on reconstructed trajectories and joint motion constraint, the extrinsic calibration algorithm is able to estimate the relative pose between the cameras.
We test the developed algorithms both in simulation and experimentally.
After trying a number of procedures, that did not guarantee an optimal result, we were able to demonstrate through the validation that the methods leads to good calibration results and that therefore, if we know the trajectory accomplished by a robot, we can estimate the relative pose of the two cameras.
The thesis is structured in this way:

- The first chapter deals with the multi-camera vision system applied to mobile robots. We will discuss about the state of the art of this kind of systems, the importance of calibrating systems, talking about the calibration in general and the various methods. Finally, we will talk about some sensors needed to collect images and data.

- The second chapter is entirely dedicated to the extrinsic calibration procedures. The method for a non-overlapping camera network is reported. Some

algorithms are explained in detail, by differentiating methods using a mobile camera from methods that do not use a mobile camera.

- The third chapter shows some methodologies and instruments to better understand the approaches developed.
- The fourth chapter describes the different algorithms developed to achieve the desired result. So, the algorithms are introduced and demonstrated theoretically, also explaining the reason for the choice of different procedures.
- The fifth chapter reports the tests carried out to validate the algorithms. The physical system used is described and shown, together with a further technical explanation on the cameras used. Finally, the results of the experiments are also presented.
- The sixth and final chapter summarizes the main considerations that were made throughout the thesis.

Acknowledgements

I would first like to thank my supervisor, Professor Alessandro Rizzo, which gave me the opportunity to do this thesis at the Consiglio Nazionale delle Ricerche and that allowed me to expand my knowledge in the field of calibration, of sensors and more.
I would like to thank Antonio Petitti, who supported and helped me in these months of work. I would like to thank him for his availability and for having guided and accompanied me during the phases of experimentation and of writing the thesis. Despite "virtual" and remote work, I learned so much from his experience.

Contents

List of Tables

List of Figures

Chapter 1

Calibration, sensors and multi-camera vision system

Nowadays, robots have spread in many industrial production sectors, even in the agri-food one. We talk about Agriculture 4.0, used to improve the quality of work, as we can see in Figure 1.1. In fact, in many farms, drones or robots are used

Figure 1.1: Agriculture by robots

to automate certain human operations, also relieving them from dangerous tasks [1]. Year by year, agricultural equipment is grown with new sensors, software that collects data using machine learning and robots walking the ground with wheels or on rails. The most innovative solutions seek to optimize processes and reduce costs, by combining robots, sensors, cameras and algorithms of satellite imaging and machine learning [2].
Another innovation comes from Denmark and it is a robot, *Farmdroid*, powered by the sun and easily transportable [3], as shown in Figure 1.2. The advent of these robots has brought a great innovation in every field of the production. Moreover, the use of mobile robots and mobile manipulators creates the flexible and autonomous industrial automation, where the greatest advantage is the exchange of information

Figure 1.2: Farmdroid

made possible by the integration of the latest intelligent technologies into robotics, such as the Internet of Things, Artificial Intelligence, or Big Data. Mobile robots thanks to all their advantages, allow to create more efficient industrial processes with a better use of resources.
The main characteristic of mobile robots is the *intelligent automation*, that means that they can perform a task without the human intervention and they are capable to make decisions. So, they can execute dangerous processes alone. The *connectivity* allows to make a communication between different machines. In this way robots can interact with humans through integrated interfaces that simplify the collaborative work. The *flexibility* is the adaptability that robots have to modify their disposal, so modifying their way of working, according to the demands of the production line, or the changes in the working environment [4].
In this thesis, we will deal with mobile robots in the field of multi-camera visual system. It is however important to underline that the calibration is fundamental in this systems, because it allows to improve the accuracy of the final result.

1.1 Calibration

Calibration is the operation in which a measuring instrument is adjusted to improve accuracy. In particular, the camera calibration has an important role in computer vision. In order to make this operation, it is important to obtain the intrinsic and the extrinsic parameters of cameras.
In the computer vision, the geometric information of the three-dimensional object is obtained from the image information of the camera. The camera calibration is responsible for providing the relationship between the spatial point and the camera image pixel. This relationship is given by the geometric model of the camera imaging, that are exactly the camera parameters. These parameters must be obtained through the experiment and the calculation and this process is the camera calibration. There is not a single useful method, but depending on applications and efficiency, different procedures are used.

Moreover, camera calibration has several applications and it is used in various areas. The following are the main areas of interest.
Autonomous navigation is a fundamental capability for a mobile robot in order to make it able to interact with the outside world and to have an interaction between the robot and the environment. In order to obtain a large scene, it needs to layout multiple cameras in the environment. So the camera calibration method has an important role in navigation and positioning accuracy. In Figure 1.3 is shown a mobile robot navigation using heading and waypoint information.

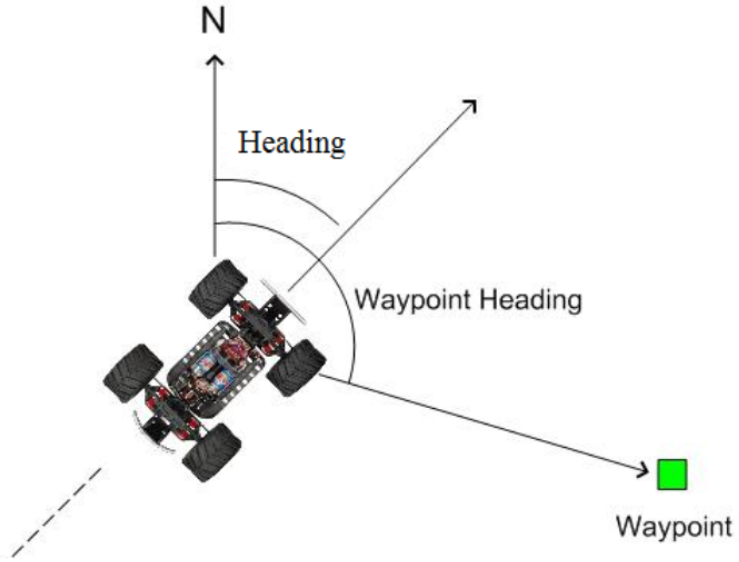

Figure 1.3: Mobile robot navigation

In the *machine vision*, the calibration methods are required to satisfy specific demands. In fact, with the development of processing technology, there is a greater demand in the mechanical field of high speed, high accuracy and high precision process automation. Machine vision can be incorporated in robotics, giving them the skills of object detection to allow for identification and the classification of numerous objects simultaneously. In Figure 1.4, we can see an application of machine vision. It also helps robots collaborate with human workers, and to integrate information from visual sources with that coming in from different sensors. This integration can help robots understand their location in space. These benefits have driven the applications of machine vision in robots [5].
In the *biomedical area*, with the help of robot and computer, it is now possible to improve the quality and the precision of the surgical operations. Moreover, thanks to robots, we can also complete the surgery and complex diagnosis which is very difficult for the routine method. It is demonstrated that with the arrival of new technologies, there is an improve of the efficiency, of the quality of the operations and there is a reduction of costs. Obviously, to make possible this coexistence, the

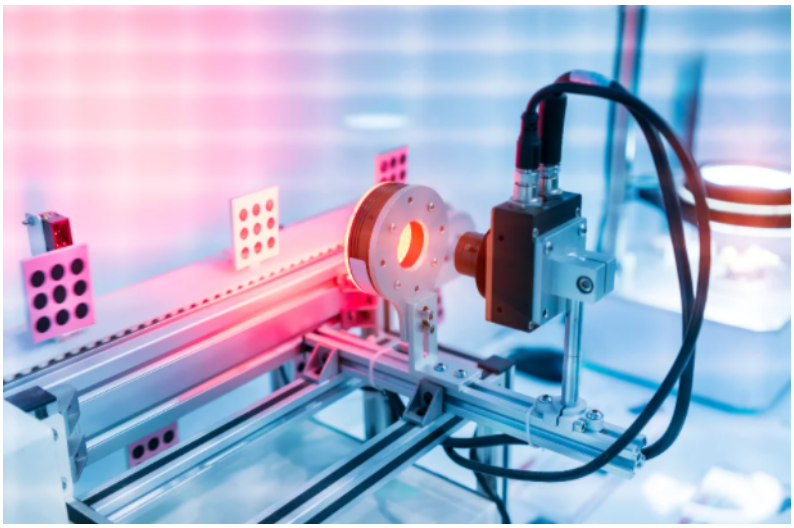

Figure 1.4: Machine Vision System

calibration of the robots and of the cameras is necessary. In Figure 1.5 we can see how robots assist doctors in performing operations.

Figure 1.5: Robots in medicine

Talking about the *visual surveillance*, we find calibration methods during the image acquisition phase. In fact, since the complexity of the road condition and vehicles are more and faster, it is necessary to improve the calibration accuracy of the image acquisition devices and the image processing speed [6]. An example of traffic monitoring system in Figure 1.6.

1.2 Why calibrate the instruments?

It is important to calibrate the instruments in order to fit together different measurements and, moreover, it allows to determine the errors and the uncertainties

Figure 1.6: Traffic monitoring system

associated with an instrument. Nowadays, there are different methods of measurement, which, however, must produce not only the best measurements, but principally the measurements with the correct meaning. The best measurements require a link to national standards, hence the need for calibration. The calibration of the instruments builds a professional credibility, because involves high quality measurements and connected to the national standards. Furthermore, it allows to detect errors and uncertainties. Finally, it permits measurements from different instruments and different instrument types to be integrated [7].
All the measurements that, then, are fit together, come from different sensors.

1.2.1 Sensors

Sensor are used to make interactions of a complex system or a robot with the surrounding environment.
In fact, "a sensor is a device that detects changes in the surrounding environment and responds by means of some outputs on another system" [16]. So, they collect external stimuli and send them to a microprocessor
There are many types of sensors, classified according to their operating principle, to the type of output signal, but mostly to the type of physical size they measure. Below we report only some.
The *light sensors* [17], whose module we see in Figure 1.7, measure illuminance, which can be used to measure more than the brightness of a light source. Because the illuminance decreases as the sensor moves away from a steady light, the light sensor can be used to compute relative distance from the source. They are used in different fields, also in the agricultural, in which the sunlight has an important role.

The *temperature sensors* [18] measure temperature and its variations. Depending

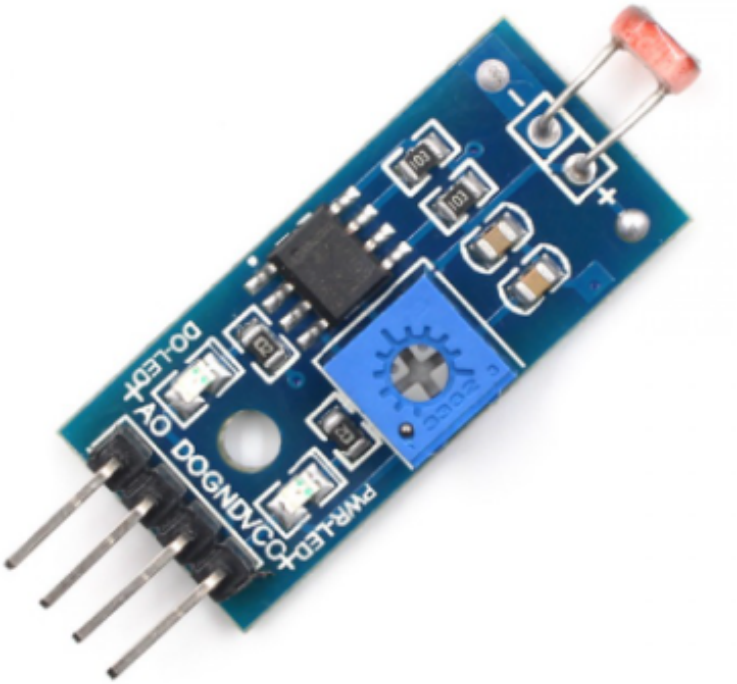

Figure 1.7: Module of a light sensor

on the applications, there are different sensors with different features. Moreover, we distinguish contact sensors, like thermocouple in Figure [1.8] and non-contact sensors, like infrared sensors, in Figure 1.9. The *optical sensors* [19], in Figure

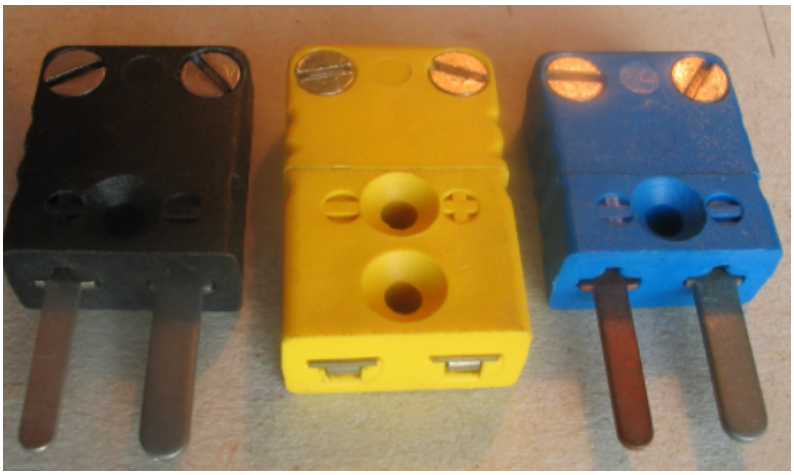

Figure 1.8: Thermocouple

Figure 1.9: Infrared sensors

1.10, convert the light rays into an electronic signal. They consist of a light source, a sensing platform, a light detector and a data processor. In the agricultural field, sensors carrying out optical measurements by means of remote sensing systems are widely used. They assess the state of the cultures based on modifications that the light radiation is affected by the plant.

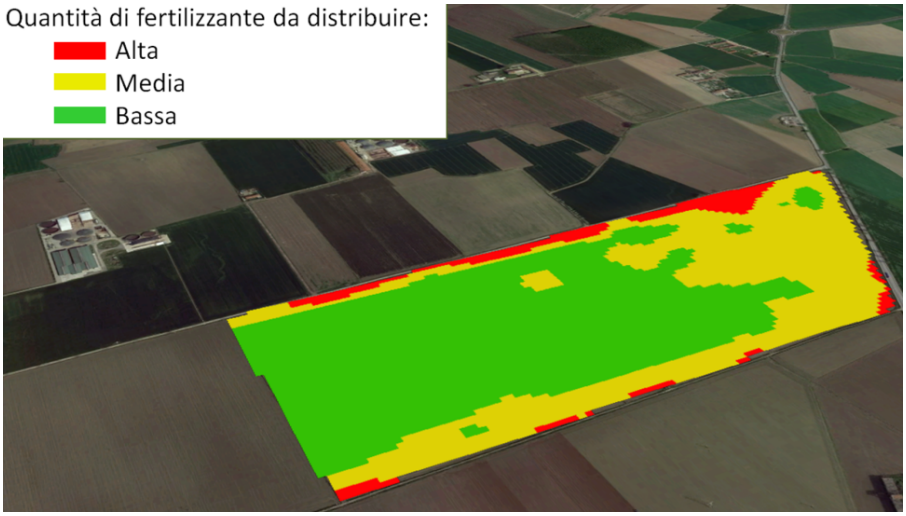

Figure 1.10: Optical sensors

1.2.2 Calibration of sensors

In order to obtain accurate and usable data, it is important to calibrate the sensors, comparing the measures of the instrument with references [20]. The main features of a sensor are the precision and the resolution.
Thanks to calibration it is possible to eliminate a possible systematic error that

will always provide a value different from the real, thus making it more accurate. In Figure 1.11 is shown how calibration can make improvements to the final result.

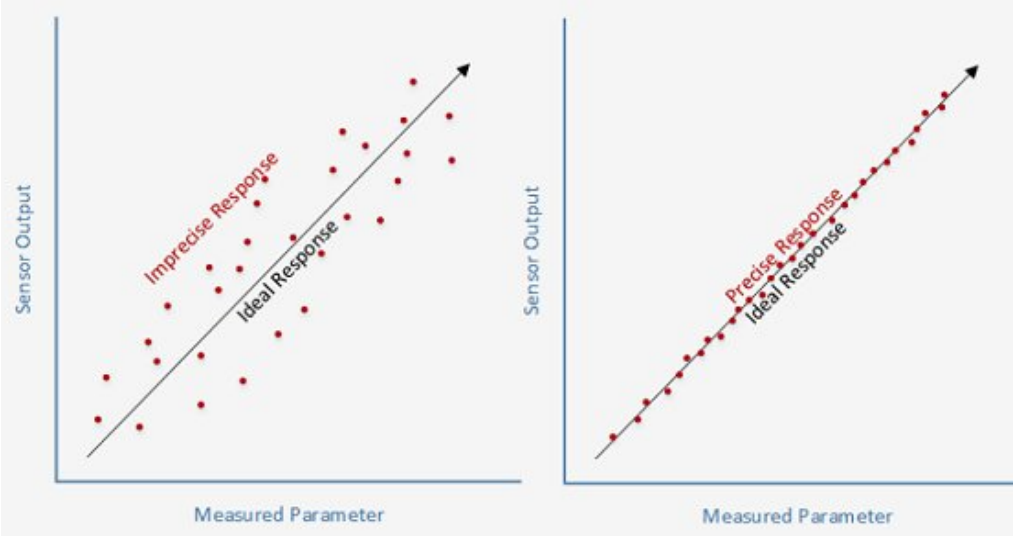

Figure 1.11: Data collected with and without calibration

Digital sensors shall be calibrated by the manufacturer, but they could still have imperfections. So, in order to obtain the best possible accuracy, it is necessary to calibrate a sensor in the system in which it will be used.
First of all, the sensor response must be compared with a reference. Later, calculate the calibration curve of the sensor, that you get after we compare, with a mathematical equation, our measurements with the standard ones.
The last step is to periodically repeat the calibration. One of the reasons why we do it is to counter the drift, which can lead to wrong conclusions.
The sensor calibration [21] deals with the estimation of the intrinsic, like the focal length, and the extrinsic, like the pose with respect to the world or to another sensor, parameters of a sensor. It is an important prerequisite for the applications in robotics, in order to fuse measurements from different sensors, because all the sensors' measurements have to be expressed in the common reference frame. While in computer vision and in the augmented reality, the pose of a sensor or of a camera with respect to the world has to be known in order to superimpose an object into the scene. In order to calibrate a sensor, it is necessary that the system is observable, so the sensors' measurements have sufficient information for estimating all degree of freedom of the calibration parameters and, given this system, it is possible to find a solution. The *intrinsic parameters* do not depend on the outside world and how the sensor is placed.
After speaking in general about calibration, its importance and its relationship with sensors, in the following sections, we will talk about different calibration methods.

1.3 Calibration in multi-camera vision systems

The system we have been working on is a *multi-camera vision system*, so in this section we will discuss about it.
A multi-camera vision system (MVS) presents some advantages respect of systems with a single camera, because they allow to have a larger field of view and to acquire geometric information of large-scale objects or scenes [8].
The global calibration of a multi-camera vision system is used to calculate the poses of the camera frames and the global coordinate frame, and also to measure the accuracy of the system. In many applications, sensors have not a common field of view, because in order to satisfy various requirements, it is necessary that cameras shoot different parts of the object.
Most of the non-overlapping field of view global calibration methods can be divided in six categories. The first one is based on *large-range measuring devices*. Then, we will talk about the *large-scale calibration targets*. Later the *optical mirrors*. The *motion model*, the *laser projection* and finally the *visual measuring devices*.

1.3.1 Methods of global calibration for non-overlapping cameras

The advantage of having multiple cameras is that we can study a larger area. However, in multi-camera vision systems, it happens that cameras have not a common field of view. In Figure 1.12, there is a system with multiple cameras with non-overlapping field of view between every two cameras.

The calibration of the system is composed of two parts. In the first moment intrinsic parameters of every camera are calibrated and subsequently, the relative position parameters between different cameras are calibrated, which is the main purpose of the calibration process.
Theoretically, each cameras' coordinate system is transformed into the world coordinate system, but, for simplicity, one camera is chosen as the main one and so, the relative pose of the other cameras is computed compared to the chosen one. When the computations have been carried out, the translation vector and the rotation matrix are computed.

Methods based on large-range measuring devices

The large-range measuring devices are high precision instruments used to ensure the accuracy of the calibration. The most common large-range measuring devices are *theodolites*, *laser trackers* and *laser range finder*.
The first method is composed of a calibration target into the field of view of the camera and two theodolites, that we can see in Figure 1.13 used to measure the 3D coordinates of the markers on the calibration target and then, the positional

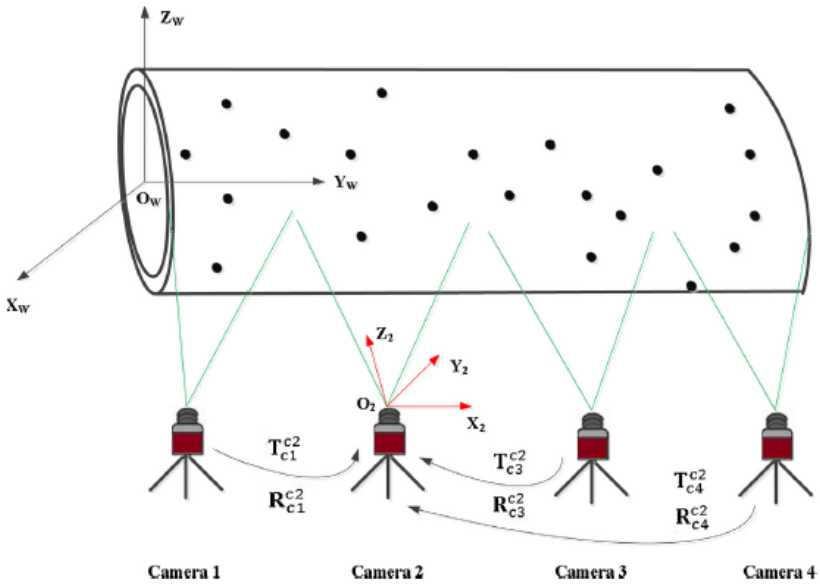

Figure 1.12: Multiple cameras with non-overlapping field of view

relationship between the calibration target and the camera is computed.

Figure 1.13: Laser with theodolite

Another approach uses the laser tracker, in Figure 1.14, which has an higher accuracy. If the corresponding relationship between the 3D coordinates and the image points of all spots on a calibration board is known, the pose of the camera relative to the laser tracker can be computed. With another strategy, a camera calibration technique for a large-scale space is presented. In this case, since there is not an enough large calibration board, a moving calibration board is used to cover the entire space. Each position of the board can be scan by laser tracker and cameras simultaneously.

Figure 1.14: Laser trackers

Another method, instead, proposes the use of external parameters calibration for multiple cameras based on laser range finder, of which we find an example in Figure 1.15. The cameras capture the laser spots projected on the planar object and the laser measures the distance between the spots. To solve the problem of non-overlapping views, the calibration of all the cameras can be achieved by pair-wise calibration. This method uses only a planar object and the laser range finder, so it is not so expensive. The advantages are that it is simple and it guarantees high accuracy, but it could be subject to certain light effects. Another method always uses a laser range finder but with an outdoor distributed camera network with a small overlapping fields of view. The data, came from the laser, are registered along the complete area of the network using SLAM algorithms. This procedure can be applied in person and robot detection localization, but it is not ensure accuracy

Figure 1.15: Laser range finder

and precision.
With all these kind of devices, 3D points of a plurality of calibration targets of a non-overlapping field of view can be obtained.

Methods based on large-scale targets

A calibration target is an instrument with high precision. Planar calibration targets are the most used calibration targets, because features are easy to extract and the place consistency is good.
The first method proposes a compound target consisting of two planar calibration targets, fixed together, as shown in Figure 1.16, in which one vision sensor is considered the global coordinate frame. The target is placed in front of the sensors for several times and the transformation matrix from the coordinate frame of each vision sensor respect of the global coordinate frame is computed. This procedure does not require high precision measurements, it has a high flexibility and it can be applied to different measurement environments. The disadvantage is that, since the distance between the cameras is greater than the field of view of each camera, the pose change of the target cannot be too large. In contrast to the 2D target,

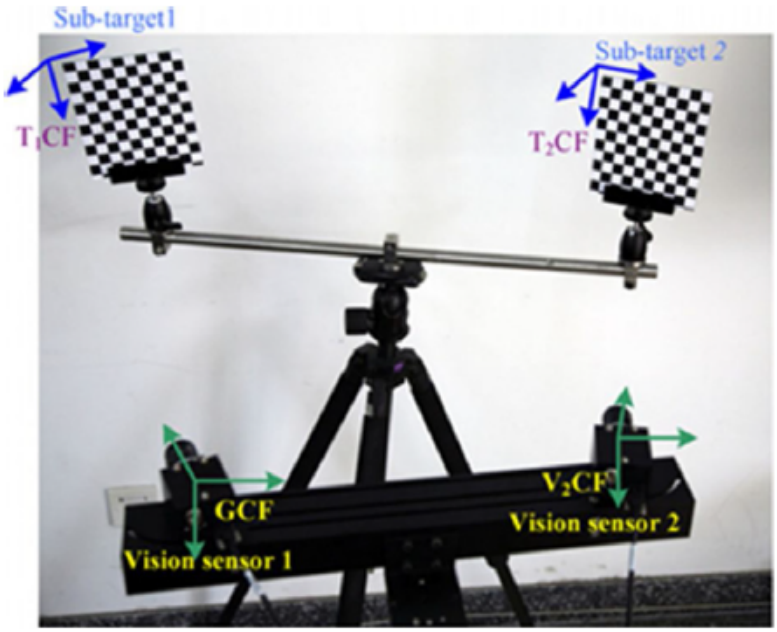

Figure 1.16: System with two planar calibration targets

1D target has a simple structure, light and portable. Another method uses, in fact, this type of target.
The cameras are divided in binocular sets. The fundamental matrix of each set is computed. In order to have a high accuracy, the length of the 1D target is proportional to the distance between sensors. Another approach starts from the estimation of the relative pose of the two cameras, which is computed changing the distances between light spots on the two 1D bars. This method is more flexible than the others. The disadvantage is that if there are more cameras, only two cameras can be calibrated at a time, which make calibration method complicated to be realized.

Methods based on optical mirrors

Mirror-based methods use the reflection characteristics of the optical mirrors. The great advantage is that the cameras do not have to see the target directly, but they can observe it through the mirror, as shown in Figure 1.17.
With regard to the case in which the camera has not a direct view of the object, the first method uses a planar mirror to estimate the target's pose [12]. The reflected views are considered as if they were seen directly. This procedure computes the camera pose and then it gets the pose of virtual camera relative to the object, estimating then the positions of the mirrors. In Figure 1.18, a calibration with a planar mirror is shown [9].

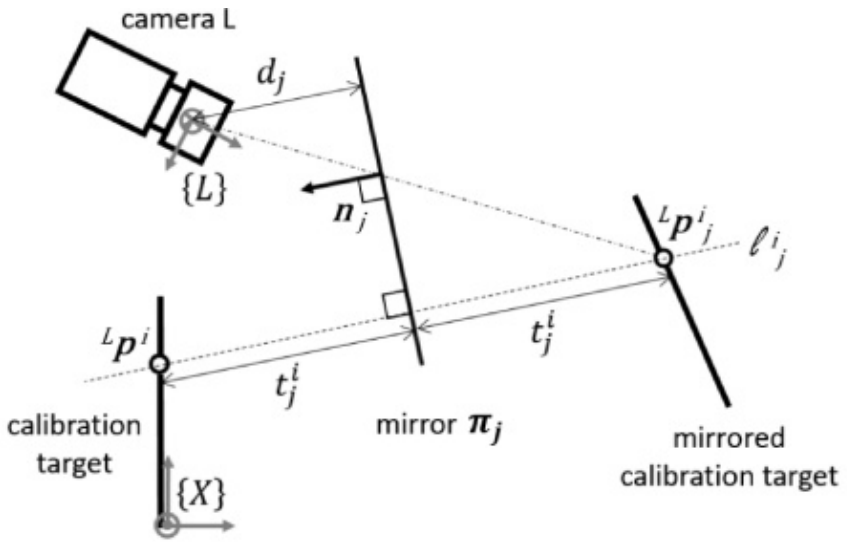

Figure 1.17: System with a no direct view

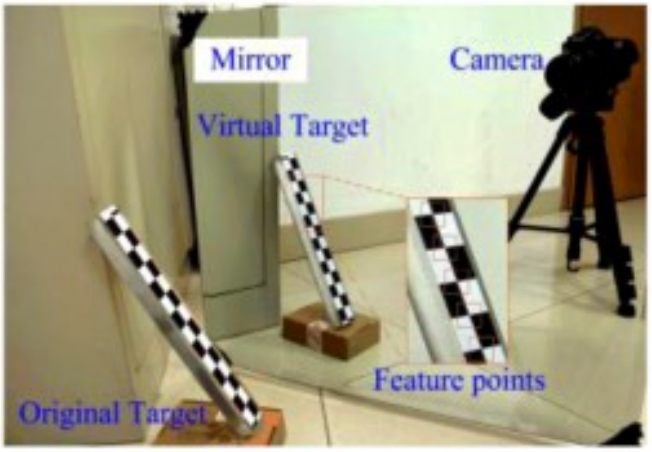

Figure 1.18: Calibration with a planar mirror

Another procedure estimates the six degrees of freedom transformation between a camera and the body of the robot, using the planar mirror to compute the calibration between them. The robot moves in front of the mirror and all the measures are collected in an estimator, which produces estimates for the camera to the robot transformation. The advantage is that is so easy to use, but it requires that the coordinates of the points on the robot-body frame must be known a priori.
The following method uses planar calibration plates and planar mirrors. It computes intrinsic and extrinsic parameters, solving a set of transformation relationship between the other cameras, including the mirrored camera and the global central camera.

Also spherical mirrors, whose radius is known, are used. The position of the mirror and the external parameters are computed in the same time. Moreover, the boundary of the mirror is visible in the image, by using the projection of the 3D reference point. So, the method is simple and practical.
These procedures are used also for visual navigation purpose in urban environment. They do not use the scene geometry, but visual markers stuck on the mirror surface. Therefore, for small systems with a small number of cameras, these approaches are simple to use. But, if the distance from the camera increases, the image of the reference object becomes smaller and the accuracy decreases.

Methods based on structure from motion

Structure from motion (SFM) is used to calibrate multiple cameras. Tracking the movement of people or other objects on the ground, this method computes the relationship of different camera field of view.
A procedure proposes the use of a rigid link [33]. It uses two internally calibrated non-overlapping cameras to capture the sequence of images in order to calibrate the external parameters of the cameras. Each camera captures a sequence, that is then processed by a SFM algorithm.
Another method assemblies cameras on a vehicle for visual navigation purpose in urban environments [10]. The calibration procedure consists in manoeuvring the vehicle while each camera observes a static scene. Then another approach considers the missing trajectory information in the unobserved areas of the multi-sensor configuration using both parametric and non-parametric algorithms. First of all, the Kalman filter is used to estimate the trajectory. Then, linear regression computes the position of the target. At the end, the relative orientation of the sensors is calculated using the target position from the adjacent cameras.
The last proposed method [11] tries to solve the problem for non-overlapping surveillance cameras with a known gravity vector. In this way, it is possible to deal with the problem of missing points correspondences required by SFM algorithms.
All these methods based on structure from motion do not require high precision devices, so they are flexible and not expensive. The disadvantage is that they need scene information, feature difficult to obtain in industrial measurements and, moreover, the accuracy has to be improved.

Methods based on laser projection

The laser projection method [13] is the most used method in the global calibration, because it has a larger measurement range and it is less affected by light. It is widely used for the non-overlapping field of view of the calibration parameters thanks to its laser point, that is small, portable and accurate.
In the first approach, the field of view of a non-overlapping camera is realized by passing the laser beam through the field of view of different cameras. It is simple,

low cost for large-scale and it is operable in narrow space environments. However, it is difficult to ensure that each camera captures high-quality planar targets and spot images.
Another procedure uses a laser pointer on a calibration target, in order to calibrate the camera pointing toward and away from the camera. The advantage is that, as long as the laser can pass through, even a narrow space, this procedure can be used. The problem is that is not sensitive to a variety of lightning conditions.
Instead, using oblique lights as target, it is possible to generate a group of laser pointers as the calibration targets. This procedure is more flexible, but it is difficult in practice, because it requires a large number of cameras. Compared with the laser line, the laser plane gives more information, increased the accuracy and it is more flexible and suitable for field calibration.

Methods based on visual measuring instruments

The vision-based measurement devices, such as *close-range photogrammetry* and *hand-held scanners*, are used for techniques of 3D reverse engineering. These techniques are employed when it is necessary to compute only the camera's external parameters, because the spatial coordinates of multiple feature points and the internal parameters are already known.
With the close-range photogrammetry, encoded targets for multi-view and unordered image sets are used together because there are not enough feature points on the surface of an object. A proposed method does not calibrate the target or the camera. In fact, it reconstructs the three dimensional coordinate points of the targets directly with a hand-held camera and then it calibrates the camera's internal parameters. At the end, each camera uses an image to estimate the external parameters, as we can see in Figure 1.19. Another approach uses a calibrator similar

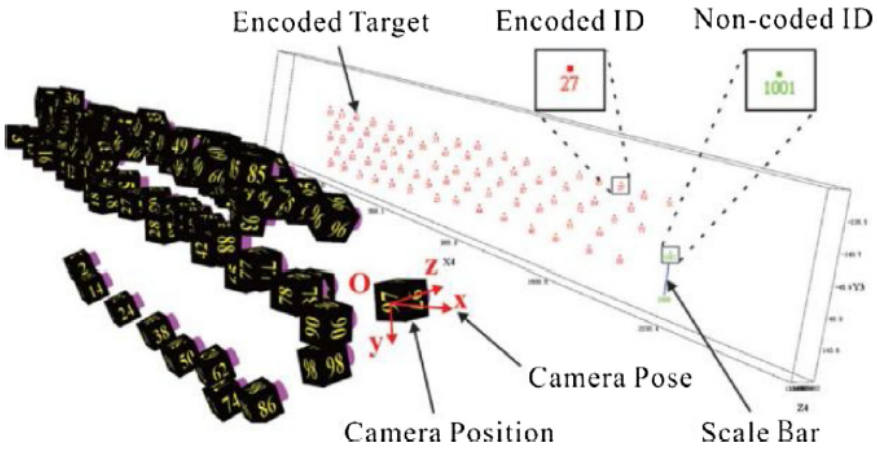

Figure 1.19: System based on close-range photogrammetry

to the shape of the measured object. So, it attaches some reflective markings on it using the close-range measurement method and then it obtains the 3D coordinates of the markers, that are applied to the camera's global calibration. This method can provide high accuracy.
Another procedure uses the hand-held scanner. With it, the measured object can be used directly as a calibrator without the need of other calibration targets. Obviously, this method is flexible and suitable for indoor or outdoor measurements.

Hereinafter, we compare the different methods, highlighting advantaged and disadvantages.
Methods based on *large-range measuring devices* have a large operating range, so it provides high accuracy. Moreover, they have high precision and for this, they are used for the geometric parameter measurement of large objects. But, due to the need of special devices, they are expensive and heavy.
Methods based on *large-scale targets* have low costs and moderate precision, so they are a trade-off in terms of costs and accuracy due to the need of simple targets. However, they require complex algorithms.
Methods based on *optical mirror* need only a mirror and a calibration target, so the cost is not so high. But, the area of calibration target in the mirror and the constrained placement of the targets can deteriorate the accuracy.
Methods based on *structure from motion* do not require high precision devices, so the cost is not high. But they need support of the scene information, that is a difficult task for the industrial and they can affect the accuracy.
Methods based on *laser projection* provide accuracy but they are dangerous for people.
Methods based on *visual measuring instruments* have high precision and good flexibility, but the utilized devices are not so expensive.

After seeing an overview of the different calibration methods, in the chapter [1.3.1] we deal in detail with the extrinsic calibration and some applications.

Chapter 2

Extrinsic Calibration Procedures

The term *calibration* of cameras meas the process of determining the parameters defining the camera model. Calibration is a process necessary to derive metric information from the image and to be able to use a camera as a measuring instrument.
In particular, the extrinsic calibration is necessary when we want to acquire a subject with two or more cameras and then reconstruct in three dimensions. Depending on your position each room will have a certain frame of the object and so the recorded images will be different from each other [14].
There is no standard imposing a particular calibration methodology, but based on the application in which the cameras are used, one technique may be used rather than another.
For this reason, in this chapter we will deal in detail with extrinsic calibration and of various methods, since the thesis handles with finding the relative pose of a multi-camera systems using extrinsic calibration methods. Thanks to this general summary, we are then able to choose the most appropriate method and that best suits our canons.

2.1 Extrinsic Calibration

The extrinsic parameters describe the pose of a sensor or a camera with respect to an external reference frame. In this scenario, the problem to estimate these parameters is called *global localization*, shown in Figure 2.1 and it is solved with efficient algorithms. Another problem could be the 3D camera localization, also known as *extrinsic camera calibration*, in Figure 2.2. If there are several sensors rigidly attached to the same device, it is important to fuse them in order to ensure that the system is observable and to increase the robustness. Fusion can be applied

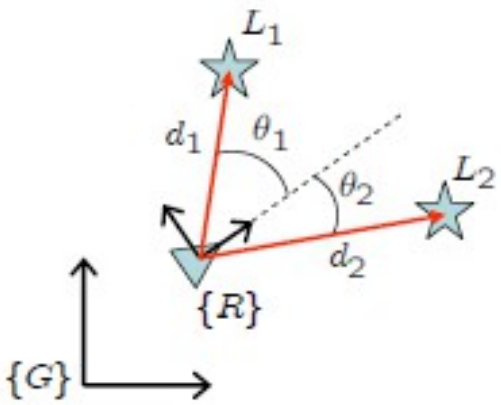

Figure 2.1: Pose of a sensor with respect to the global reference frame

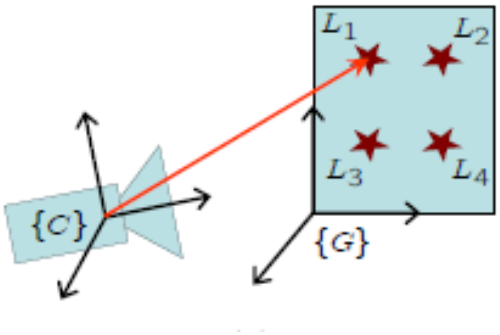

Figure 2.2: Pose of a sensor in the 3D space

only if all the sensors are spatially related. These problem is called *sensor-to-sensor transformation* and it deals with expressing all the measurements with respect to a common reference frame. The process of estimating this transformation is known as *extrinsic sensor-to-sensor calibration* and, depending on the type of sensor used, is divided in *pairs of sensors whose spatial measurements can be correlated*, in which the sensors can localize themselves with respect to a common reference frame and *pairs of sensors whose spatial measurements cannot be directly correlated*, in which the pose of two sensors cannot be obtained with respect to the common reference frame and so, considering that they are rigidly connected, we can deduce their transformation [21].
The extrinsic calibration is used for data fusion, so, as we have already said, point clouds that come from different sensors are studied into a common reference frame [22].
Below, there is a quick roundup of different methods of extrinsic calibration and

between different devices.
The following sections will occupy of the application of the extrinsic calibration using different sensors or cameras.

2.2 Extrinsic Calibration of a camera and laser range finder

Let us consider now a method for extrinsic calibration of a camera and laser range finder.
Two dimensional laser range finders mounted on a mobile robot become very common in the navigation task field. In fact, they provide in real time accurate range measurements and enable robot to perform tasks by fusing image data from the camera mounted on them. In order to use these data, it is important to know the position and the orientation of the camera with respect to the laser range finder. The calibration is divided into internal and external parameters. The latter are the position and the orientation of the sensor relative to a coordinate system.
The method of extrinsic calibration we will talk about in this section uses a planar calibration pattern viewed in the same time by the camera and the laser range finder.
As we can see in Figure 2.3, there is our system in front of a checkerboard, so a planar pattern, that is visible to both the camera and the laser range finder. First

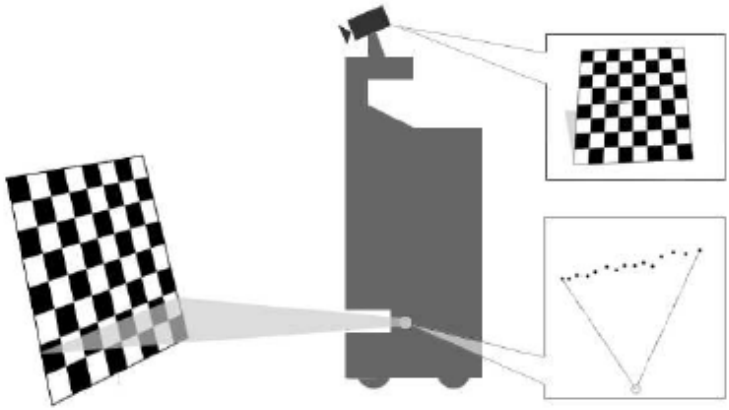

Figure 2.3: A planar calibration pattern is posed in the both views of the camera and the laser range finder

of all, we propose a linear solution to solve the problem. Then, we will consider a nonlinear optimization with outlier detection and at the end, we will talk about a global optimization. The latter is done to refine both the intrinsic and the extrinsic parameters more accurately.

With the first solution, so the linear one, we assume that the camera is calibrated and we need to know the calibration plane parameters by solving the pose of the camera. Initially, we determine the camera's extrinsic parameters and then, we obtain the calibration plane parameters. For each pose of the calibration plane, we will have several linear equations, that we can solve with linear least squares. The second approach, which is the nonlinear optimization, proposes to minimize the distance between the laser points and the checkerboard planes. While, with the last procedure, the global optimization, the camera calibration is not known and the measurement errors can affect the final result. So, given the orientation and the position of the camera, the laser gives some constraints on the position of the planar pattern, that are analyzed in the camera intrinsic calibration. In conclusion, to refine the camera intrinsic and extrinsic parameters, it is appropriate the global optimization with the initial estimate of the both intrinsic and extrinsic parameters. For further clarifications, please consult [24].

2.3 Extrinsic Calibration of a 3D-Lidar and a camera

In this section, we will present an extrinsic parameter estimation algorithm between a 3D LIDAR and a projective camera. This method uses data coming from the common field of view of the LIDAR and the camera, placing the planar board at different poses.
First of all, the procedure localizes the target and its edges in LIDAR and camera frames. Then, it matches planes and lines and at the end, it solves a cost function composed of the geometric constraints that links the features detected in both the LIDAR and the camera with non linear least squares.
Cameras and LIDARs work complementarily. In fact, depth information that cameras cannot provide, are provided by the LIDARs. In the same way, color, texture and appearance information that LIDARs cannot give, are given by the cameras. Furthermore, cameras can recognize objects, but it cannot tell us how far it is, information that LIDARs give us. Camera sensors are better then LIDAR sensors for place recognition and loop closures, but they are affected by illumination. Instead, LIDAR sensors are immune to illumination changes.
So, cameras and LIDARs are used together for multi-sensor state estimation, mapping and localization and, not by chance, the Simultaneous Localization and Mapping (SLAM) algorithms need both cameras and LIDARs for robust state estimation.
In order to fuse the information coming from these two sensors, it is necessary using the extrinsic calibration procedure. Most algorithms assume the extrinsic calibration to be known a-prior, so it is important to estimate the Euclidean 6 Degrees of Freedom (DoF) transformation between a LIDAR center and a camera center

using data generated by them.
The 3D-LIDAR camera extrinsic calibration problem is classified into two categories,*target based approaches* and *target less approaches*. Moreover the latter is divided into *scene based approaches* and *motion based approaches*. Target less approaches rely on a good initial guess and are sensitive to the calibration environment. While, the target based approaches are computationally light weight and offer good initialization for target less approaches, if available. So, they simplify the data association problems.
The resolution of this problem is based on the technique described in the previous section, i.e. the observation of a planar checkerboard by both the sensors. The checkerboard gives the pose of the planar surface in the camera frame. The extrinsic parameters between the camera and the LIDAR are determined by solving the constraint formed by projecting the LIDAR points on the checkerboard plane in the camera coordinate system. In [25] are explained, in detail, the different procedures.

2.4 Extrinsic Calibration and odometry for camera-LIDAR systems

Continuing to talk about the camera-LIDAR systems, in this paragraph we focus on another method, precisely, a two-stage extrinsic calibration method as well as a hybrid-residual-based odometry approach. The extrinsic calibration can estimate the relative transformation between the camera and the LIDAR with high accuracy, in order to register image and point cloud data in a proper way. After the calibration, the hybrid-residual-based-odometry can be used to provide real-time estimates. This approach exploits both direct and indirect image features.
The problem to be studied is the camera-LIDAR extrinsic calibration problem so that the LIDAR and the camera data can be registered under a common reference frame. As we have already said, the proposed method is a two-stage extrinsic calibration method. With the first calibration stage we obtain a proper initial guess of the extrinsic parameters by using the constraints between the motions of individual sensors. With the second stage we refine the results by registering the LIDAR information to the image information, assuming the camera and the LIDAR have a sufficiently overlapped field of view.
In the previous sections, we discuss about methods based on motion-based approaches and mutual-information-based approaches. Both presented disadvantages and advantages, so in order to combine these latter, the two-stage calibration method is developed. In fact, the first one can work without an initial guess, but its estimation is not so accurate. While, the other one, can provide more accurate calibration results, but it needs an initial guess of the extrinsic calibration.
We assume that the camera and the LIDAR are synchronized and the intrinsic calibration parameters are pre-calibrated. The extrinsic parameters are the rotations

and the translations of the LIDAR coordinate frame with respect to the camera coordinate frame. Detailed explanations of the two methods, are cited in[26].
After the calibration, to guarantee an accurate real-time motion estimation, it is proposed an hybrid-residual-based camera-LIDAR odometry method. The problem found in this approach is similar to the problem of estimating the camera motion using both the image and the points cloud data, i.e. due to the limited number of laser beams in the LIDAR, the point clouds are sparse in the vertical direction, and this can pose difficulties in the registration. On the other hand, the camera images give dense appearance information, which means high costs. So the solution is the combination of the two.
Instead in this case, our approach relies on two types of the landmark, the *photometric landmarks* and the *reprojection landmarks*. In Figure 2.4 is shown an example of extracted photometric landmarks (marked green), on the left and an example of extracted reprojection landmarks (green mark) on the right. We assign each of the

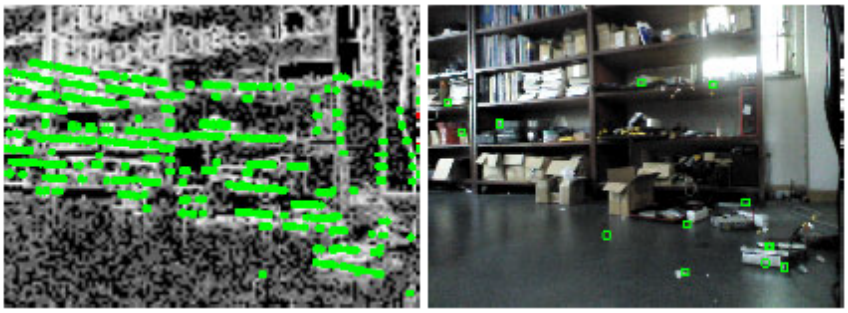

Figure 2.4: Photometric landmarks and Reprojection landmarks

landmarks either from a LIDAR range readings and through a depth interpolation algorithm. Each landmark introduces a photometric or a reprojection residual term in the motion estimation model. Both models have been clearly explained in [26].

2.5 Extrinsic Calibration with a thermal camera

In this section, we deal with another method, i.e. the extrinsic calibration between a sparse 3D LIDAR and a thermal camera, using a monocular visual camera [23].
All the sensors have limitations. In fact, not all sensors guarantee the same results in all environments. For this reason, robots are equipped with multiple sensors. In order to combine all the information came from them into a common reference frame, it is necessary to know the relative position and orientation between sensors, obtained thanks to the extrinsic calibration.
A 3D LIDAR is used mostly in robot devices, because they can capture distant details more that any other type of sensors. However, if the lighting condition

is too poor, the device is disable. So, in order to compensate this lack, thermal cameras are used. Since a direct method to calibrate a sparse 3D LIDAR with a thermal camera does not exist, we will consider a two-step method to obtain the extrinsic calibration. First of all, we obtain the transformation matrix between the LIDAR and the visual camera. Then, we obtain the transformation matrix between the thermal camera and the visual camera. And at the end, the extrinsic parameters are calculated by multiplying the above two matrices.
The first proposed approach uses a rotating 2D LIDAR to produce 3D point cloud. In order to calibrate the device with a thermal camera, a checkerboard with black and white melamine blocks is utilized. However, the thermal contrast is not good to detect the corners. So, another research proposes a board with a large circular hole, that is easier to detect by configuring the background temperature. Another method deals with the extrinsic calibration between a visual camera and a thermal camera. In this case, we have a checkerboard with square holes. The calibration is done only if the image domain is similar to a stereo calibration.
In conclusion, the extrinsic calibration between a 3D LIDAR and a thermal camera can be done using a checkerboard with circular or square holes. Comparing different kind of checkerboards, it has been noted that a board with asymmetric circular holes, in Figure 2.5, fits better with the problem [27].

Figure 2.5: Board with asymmetric circular holes

2.6 Extrinsic Calibration of the non-overlapping camera network

Now let us focus on a method for a non-overlapping camera network, based on close-range photogrammetry. In Figure 2.6 there is a diagram of the field of view of a non-overlapping camera network. The procedure consists of three steps. The first one is reconstructing the 3D coordinated of the target. Then it performs the intrinsic calibration and, at the end, it calibrates the extrinsic parameters of each camera. Multiple cameras are used because they are more convenient respect of the single-camera systems, since they can be employ to cover a wider field of view.

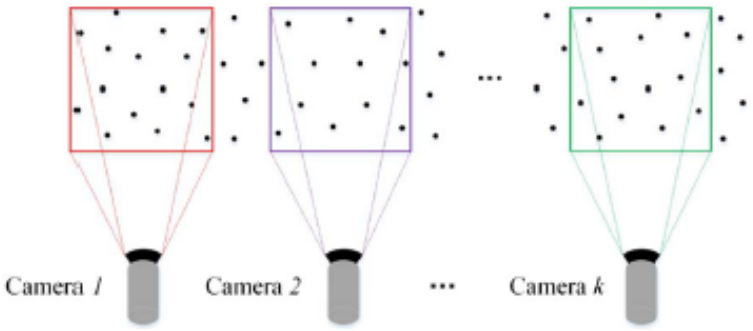

Figure 2.6: Diagram of the field of view of a non-overlapping camera network

The applied algorithms for the extrinsic calibration of non-overlapping cameras can be classified into two categories, those in which the camera has may not move and those in which the camera has to be moved.
Speaking of the first category, a proposed approach uses a moving object in the cameras' field of view to compute the poses of the camera and to build the trajectory. A suggested way uses a maximum a posterior (MAP) estimation, with which the target moves varying velocities and directions, but it is assumed that the target cannot do sharp turns. A more robust solution allows to the target to make sharp turns along a trajectory. Another studio has dealt with improving the robustness and reducing the amount of prior information, using the unobserved trajectory and estimating the position and orientation, at the expense, however, of the accuracy of the method. Recently, another research proposes to calibrate the extrinsic parameters of multiple vision sensors based on a 1D target. The problem is that, even if the 1D target has a simple structure, it would bend or deform and this can affect the final results.
In order to have an high accuracy, other methods were developed. The first one is the photogrammetry method, that adopts a set of encoded targets to obtain the relative poses of the cameras. The calibration procedure is divided in three stages: the reconstruction of the 3D coordinates of the targets through an hand-held digital camera, the calibration of the intrinsic parameters and the localization of each camera with only one image. With this procedure, the extrinsic calibration is done very quickly, so it is more efficient than the other methods.
While, speaking of the methods that need a moving camera, the two main solutions are the *hand-eye calibration (HEC)* and the *mirror-based method*. As regards the first research, it is thought to use sequences of the poses of each camera to estimate the localization of multiple rigidly coupled cameras. Subsequently, after other attempts to improve the accuracy, a solution is provided for calibrating both intrinsic and extrinsic parameters of the non-overlapping camera rig simultaneously. Referring to the mirror-based method, the mirror has been used to generate an overlapping view between cameras, which moved a mirror pose camera instead

of the real camera. This procedure is not so convenient to implement because it requires a mirror during the calibration.

2.7 What is the best way?

In conclusion, we can say that, for estimating the extrinsic parameters of the camera network is necessary to determine the localization of each camera relative to the world coordinate system, that has to be determined, while the 3D coordinates of all the points have to be obtained. Before starting the calibration, the intrinsic parameters are required, through one of the methods above mentioned. Significant assumptions are that the cameras are synchronized and the estimation of the pose of a calibrated camera is achieved from 3D to 2D point correspondences. So, the localization of all the cameras has been determined and, hence, it is easy to obtain the relationship between each camera. In Figure 2.7 there is a description of the transformation between cameras in non-overlapping camera network.

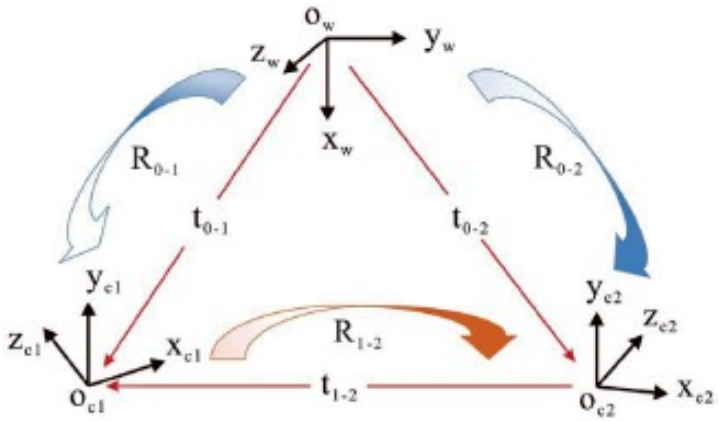

Figure 2.7: Transformation between cameras in non-overlapping camera network

Since our system uses moving cameras with non-overlapping field of view, the most suitable procedure is the one just described, that is the *Hand-eye Calibration*, mentioned in detail in the chapter 4.
The HEC [15] method is also called robot-sensor or robot-world calibration, because it determines the transformation between a robot and a camera or a robot and the world coordinate system. There are two kind of HEC, shown in Figure 2.8. In the first one, on the left image, the camera is mounted stationary next to the robot, as in our situation. In this case, we have to find the transformation from the camera's coordinate system to the coordinate system of the robot base. In the other one, on the right image, the camera is mounted on the robot. It is a convenient approach because it is able to improve the accuracy and to minimize the scheme using robot- and corresponding camera poses. Robot poses are read directly from the robot,

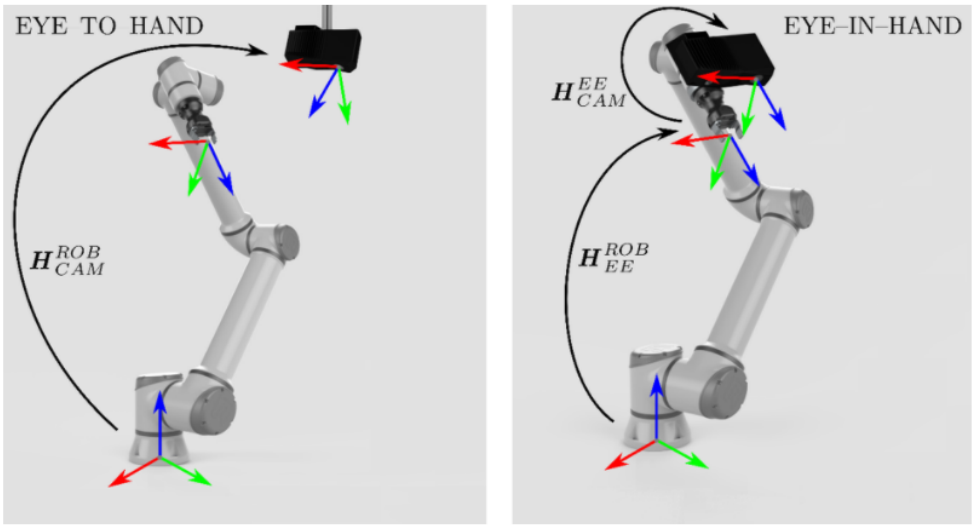

Figure 2.8: The camera mounted next to the robot and the camera mounted on the robot

while camera poses are calculated from the camera image.
In order to solve our problem, we combined the HEC method with the LM method, adapting it with modifications, explained better in the chapter 4.

Chapter 3

Methodologies and instruments

In this chapter we will talk about all the methodologies and instruments useful to better understanding the procedures introduced in the following.

3.1 Structure From Motion

The Structure From Motion (SFM) algorithm is a method for 3D reconstruction from 2D images and it is used to "determine the spatial and geometric relationship of the target through the movement of the camera" [36]. The computation of 3D reconstruction from 2D images is composed of three steps, that are *rectification*, in which a transformation of each image is determined to reduce the correspondence problem from 2D search to 1D search. The second step is the *correspondence search*, in which the correspondence between pixels is determined in the left and in the right image, that is found searching the same row of the pixels in both the images. The last step is the *reconstruction*, in which we compute th3 3D at that pixel, using the triangulation algorithm for each pixel and its correspondence. The main model to build a SFM system is shown in Figure 3.1. The relationship between the red block, the estimation of the multiple view geometry, and the blue box, the feature tracking, consists of the multiple view relationships used to regularize the feature tracking. The 3D structure is based on the features and on the estimation of the camera parameters. If the extracted features are image points, the 3D structure will be a 3D point cloud.
Furthermore, depending on the type of used data, there are two kind of approaches. The first one is the *feature-based reconstruction*, in which it uses characteristics of two images representing the same scene but from two different points of view. The other one is the *flow-based reconstruction*, that through the optical flow, studies the features velocity field generated by the camera motion[37].

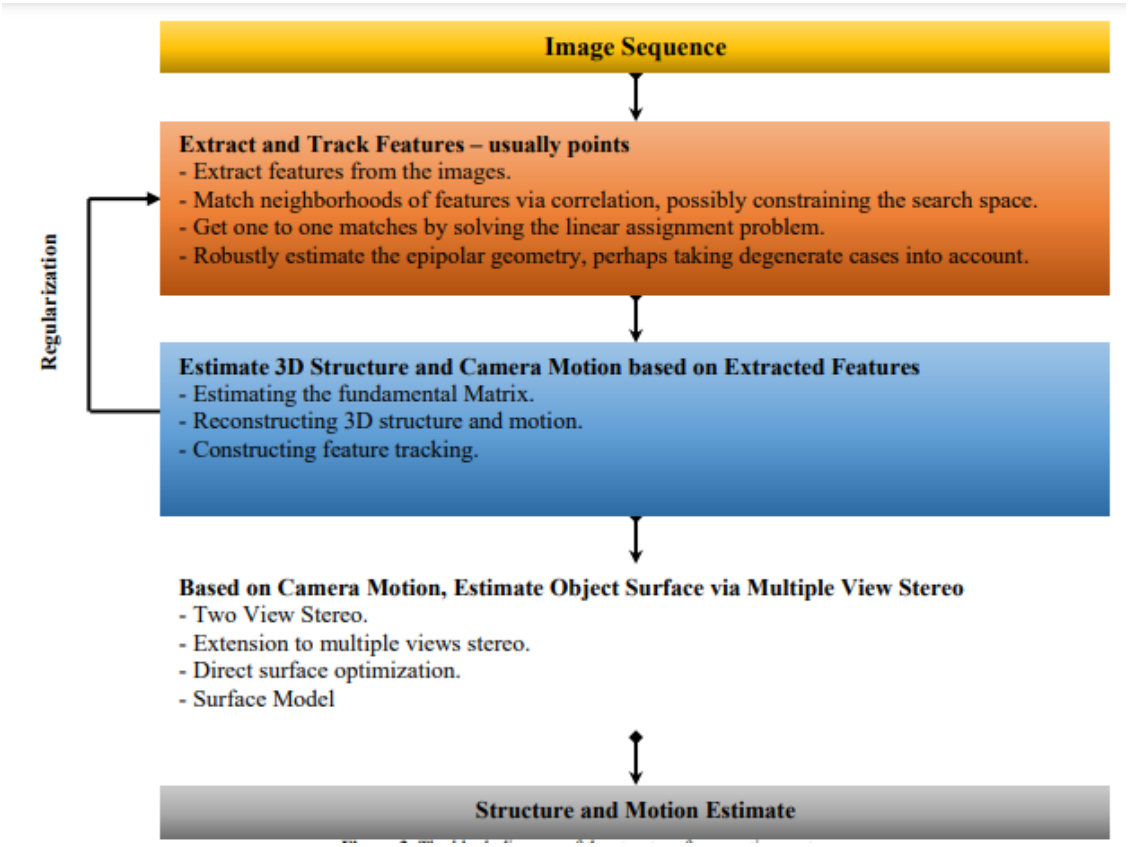

Figure 3.1: The block diagram of the SFM system

So, to reconstruct the image is necessary to identify the image points corresponding to the same 3D points of view in different images. Those pixels that are difficult to separate from one other, are removed. This is the first step, called *feature detection*. The successive step, *matching phase*, finds feature points in different images that correspond to the same 3D point. In the last step, the geometric model is built. For further clarifications, please consult [38].
The advantages of this method are that is not expensive, because it only uses a RGB camera, there is an high resolution 3D data acquisition. But it is a complex algorithm, so it needs speed and accuracy.
Nowadays, this SFM is applied in different techniques, that have been made easier thanks to this procedure. For example, in the *image-based 3D modeling*, while creating a simple 3D model was easy, obtaining an accurate 3D model of a complex real scene was difficult. To solve this problem, the SFM is used. Another application is in *hand-eye calibration*, in which, previously, a calibration object was used, that now it was replaced by SFM algorithms. In the *augmented reality*, some limitations, like to place a virtual object with correct poses in a proper location, were overcome using SFM. In the *photo organization and browsing*, the SFM resolves the problem

of estimating the viewpoints of cameras from a large number of unordered images. Other applications, less used, are explained in [39].

3.2 Iterative Closest Point

The Iterative Closest Point [35] is an algorithm used in the process of registration of 3D point cloud data to describe the environment and it is applied to minimize the difference between two clouds of points and to reconstruct 2D or 3D surfaces. It takes two point clouds as input and returns the rigid transformation. It formed by different steps. Let us consider two input point clouds S and M, where S is the source point cloud and M is the model point cloud. In the first step, we compute the nearest point in S for every point is M, using the Euclidean distance. If this distance is bigger than a threshold, we remove a given pair of points. Then, we add the weights to pairs of points and we compute the rotation matrix and the translation vector. Later, we compute the the transformation of the set S and at the end, after calculating the error, we iterate until we reach the required accuracy [40].
The tricky part is to start to an appropriate initial value and an approximate registration of two point clouds to ensure that the algorithm does not fall down into local extremes, but in the actual point cloud.
One of the methods used for the algorithm is the LiDAR technology. This procedure measures the distance to a target by illuminating it with a pulsed laser light and it measures the reflected pulse with a sensor. When the sensing device moves, the final cloud of points is partially overlaid, so it is necessary to assemble these scans together to form the final object. Neverthless, with laser scanners is impossible to obtain all the point cloud information of the object, because it has some limitations on the field of view. In order to have a complete point cloud, you have to find a 3D rigid body transformation, so that the 3D coordinates of the point cloud at different angles can be overlapped.
Furthermore, several improved algorithms based on ICP exist. One of these proposes to find in the current set, in each iteration, the nearest point of each point set at the point of the other set. With another one, the algorithm has to find the nearest point from another set point as the corresponding point of the current set. And finally, specifying the error metric function, we can improve the accuracy of point cloud registration.
For more details about the mathematical solution of the problem, see [41].

3.3 Simultaneous Localization And Mapping

SLAM is the acronym for Simultaneous Localization And Mapping. It is used to estimate the pose of a robot and the map of the environment at the same time. It is a

complex method because the map is needed for localization and the pose estimated is needed for mapping. Moreover, it is considered a hard problem because, since both the path and the map are unknown, the errors in map and pose estimates are correlated. Furthermore, since the mapping between observations and landmarks is unknown, choosing wrong data associations can make catastrophic events.
The SLAM procedure is used in air and underwater applications, and also in indoor and outdoor implementations for autonomous and unmanned vehicles [44].
There are three major models of algorithms from which all the SLAM methods derive. The first one uses the Kalman filter (EKF) for representing the robot's best estimate. There are different steps leading to build the final map, which are the *state prediction*, in which the robot-landmark cross-covariance matrix is predicted, the *measurement prediction*, in which the pose of the robot and the pose of the landmarks are estimated, the *observation*, the *data association*, in which the predicted measurements are associated with observation, the *filter update*, in which the Kalman filter is applied and at the end the *integration of new landmarks*, in which the coordinates of the landmarks are added to the state vector. The final result is shown in Figure 3.2, in which in the right image we can see the map and in the left image we can see the correlation matrix [44]. The second procedure uses the

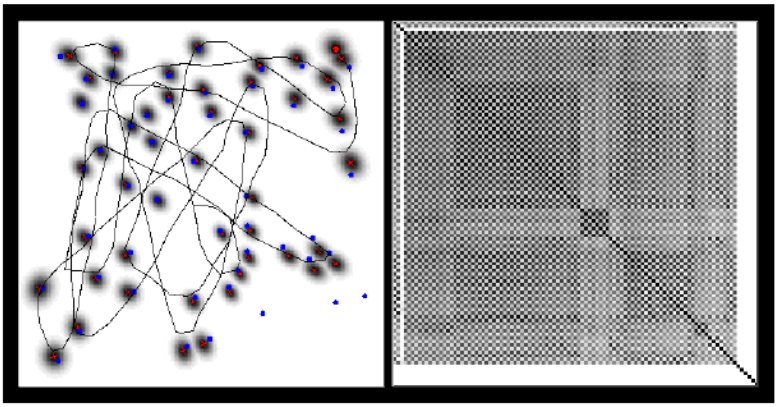

Figure 3.2: Map and Correlation Matrix

particle filter method, a method where a set of particles is used to represent robot's belief, instead of using parametric values. Each particle has an hypothesis of the robot pose that assume its pose is correct. Starting from this assumption, every particle build their map. Obviously, some particles will have more accuracy than others. The negative aspect of this process is that the number of particles required grows exponentially with the size of the space state. Even in these algorithm, there are several steps to implement it. The first one is drawing a distribution of weighted particles, then there is the update of the robot's state in each particle. Later the observation data are collected, with which the robot's state estimate is updated thanks to the robot's observation. After, the weight of each particle is calculated

and at the end the particles are resampled proportional to their weights [45].
The last model begins with the fact that the SLAM problem is like a sparse graph of constraints, so it uses nonlinear optimization for recovering the map and the robot's pose. So, whenever the robot obtains a measurement, we add a node to the graph. This node represents the pose of the robot at which the measurement was obtained.
Other fields outside robotics, for example the photogrammetry and the computer vision, have studied the making of environment models from a moving sensor platform. Starting from these works, the SLAM algorithm tried to solve the problem. There is not a single solution to the SLAM method, but the chosen method depends on a number of factors, such as the map resolution, the features of the map, the update time and so on [46].
With the SLAM, the devices take data from sensors to build a picture of the environment and to estimate their position. These sensors can use *visual data*, such as cameras, *non visible data sources*, such as Sonar, Radar or Lidar, and *basic potential data*, using an Inertial Measurement Unit (IMU). When the device moves, all the features in the environment will move in relation with it, so the SLAM algorithm can study and improve the new positional information. In fact, "the more iteration the device takes, the more accurately it can position itself within that space". So, SLAM algorithms are designed for devices that are moving through a space [47].
Let us now analyse how the SLAM algorithm works. There are two types of technology components to achieve SLAM, the *sensor signal and front end processing* and the *pose graph optimization*. Talking about the first type, we distinguish *Visual SLAM* and *Lidar SLAM*.
Visual SLAM uses images acquired from cameras or other vision sensors. It can use simple cameras, compound eye cameras and RGB-D cameras. Moreover, since cameras give a lot of informations, they can be used to detect landmarks, improving the SLAM flexibility. When the SLAM uses a single camera, it is called *monocular SLAM*.
LIDAR SLAM means Light Detection And Ranging and it uses a laser sensor or a distance sensor. Lasers are more precise than other cameras and they are used for applications with high speed moving vehicles, such drones. By matching the point cloud, the movement is estimated and it is used to localize the vehicle. The disadvantages of the point clouds are that they are not detailed as images, so they do not give sufficient informations. In addition, they require high processing power, so it is necessary to optimize the processes to improve speed. For these reasons, in order to obtain the desired effect, it is necessary to fuse other measurement results, such as wheel odometry, global navigation satellite system and IMU data.
SLAM algorithms are used for practical applications, that, however, could generate problems and errors. For example, SLAM estimates sequential movement, including some margin of error, which is accumulated in time. This is called *loop closure* problem. This type of errors are unavoidable, so it is important to detect them

and try to correct them. In order to do that, it is opportune to remember some characteristics from a previously visited place and minimize the error. Another problem in the development of the algorithm concerns the computing costs. Costs depending on several utilized processes, such as image processing, point cloud processing and optimization. In order to overcome this problem, it is possible to run different processes in parallel [48].

3.4 Real Sense T265 and Visual Inertia Odometry

The RealSense T265, shown in Figure 3.3, is a camera that " uses proprietary simultaneous location and mapping technology with visual-inertial odometry (V-SLAM)" [50]. The RealSense T265 is a surveillance camera based on visual processing unit

Figure 3.3: Real Sense T256

(VPU) and it processes all the data necessary for tracking, so it is ideal for applications where it is important to locate the location of a device [52]. The localization could be indoor or in remote places, so it is easily added to little devices like light robots or drones, but also smartphones or viewers.
It is a system inside-out, that is, it does not use external sensors to understand the surrounding environment and it offers a 6 DOF monitoring, collecting informations from two cameras.
As already mentioned before, it uses a Visual Inertial Odometry (VIO). The VIO is "the process of estimating the state of an agent by using only the input of one or more cameras plus one or more Inertial Measurement Units(IMU) attached to it" [51]. These sensors are present in almost all the robots used today, because both

cameras and IMUs are very cheap.
Unlike the SLAM algorithm, in which the feature positions and device pose are both estimated, with the VIO, pose estimation is achieved without assuming a feature map.
All of the V-SLAM algorithms run directly on the VPU, providing low latency and efficient power consumption. That is why, these devices are used in applications that require high precision and low latency.

3.5 Robot Operating System and rosbag

The Robot Operating System (ROS) is a framework and a collection of tools, libraries and conventions for the development and programming of robots. It provides the same functions as an operating system, but it is not a real-time operating system, although it is possible to integrate ROS with some real-time modules. This system is not only used for robots, but the majority of tools provided are focused on working with peripheral hardware [53].
ROS is considered a coupled system where a process is a node and every node is responsible for one task. Nodes communicate using messages passing via logical channels called topics. Moreover, each node send and get data from other node using a public model. In order to manage the coupled system, there is a master which is responsible of all the actions that take place in the system, as shown in Figure 3.4. Messages are structs of data filled with pieces of information

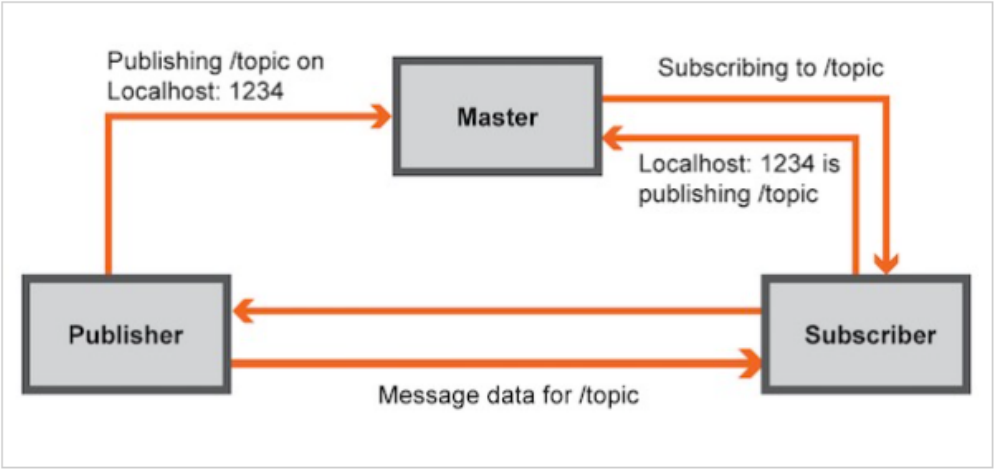

Figure 3.4: Master in ROS

by nodes [54].
The focal point is this communication system, allowing you to design complex software without knowing how certain hardware works. So, developers can assemble a

complex system by connecting existing solutions for small problems. The number of tools connected to the framework are its biggest power, in addition to reactivity and low latency. The principal advantage is that it allows collaborative development of robotics software. For example, if we consider several laboratories working on similar projects and that could interface and help each other, thanks to ROS they can collaborate and build upon each other's work [55].
Data contained in ROS messages can be recorded in specific files, called bags. *Rosbags* "is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserialization of the messages"[49]. The principal advantage is that the recording can be used several times, reproducing each time the exact operating scenario in which the bag was recorded. For example, registered messages can then loaded without the need to repeat the experiment, thus allowing to develop more easily algorithms [56].

Chapter 4

Calibration algorithms for a multi-camera system

In this chapter, we explain the methods studied and implemented.
The considered setup assumes two 3D cameras rigidly coupled moving in the space. Each camera captures, in time, a cloud of 3D points and estimates the trajectory traveled in space, through SLAM algorithms. Based on reconstructed trajectories and joint motion constraint, the extrinsic calibration algorithm is able to estimate the relative pose between the cameras.

4.1 Levenberg-Marquardt method

Given two trajectories constrained by a constant rotation matrix R and a translation vector C, the objective of the method is to estimate this constraint, which is of how much they are rotated and translated with respect to each other.
The procedure [28] proposes to find a solution that minimizes the sum of the squares of the errors between the model function and a set of data points.
We have a set of data $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, where x_i are independent variables, $(x_1 \neq x_2 \neq \ldots \neq x_N)$n y_i are the observations and N is big. We want to fit the data with a straight line $p(x) = a_1 + a_2 x$. It is not possible to find a straight line such that $p(x_i) = y_i$, so we look for the straight line that minimizes

$$S = \sum_{i=1}^{N} (y_i - p(x_i))^2 \tag{4.1}$$

Each $y_i - p(x_i)$ is a residual, that is the difference between the observed value y_i and the predicted one $p(x_i)$. Rewrite it through a matrix form, you get $Ax = y$, in

which

$$A = \begin{bmatrix} x_{11} & \dots & x_{1,N} \\ \dots & \dots & \dots. \\ x_{N1} & \dots & x_{N,N} \end{bmatrix} \tag{4.2}$$

$$x = \begin{bmatrix} a_1 \\ \dots \\ a_N \end{bmatrix} \tag{4.3}$$

is the vector of size N of the linear combination coefficients and

$$y = \begin{bmatrix} y_1 \\ \dots \\ y_N \end{bmatrix} \tag{4.4}$$

is the vector of size N of the experimental measurements.
The final result is a straight line approximating the N observations, as shown in Figure 4.1.

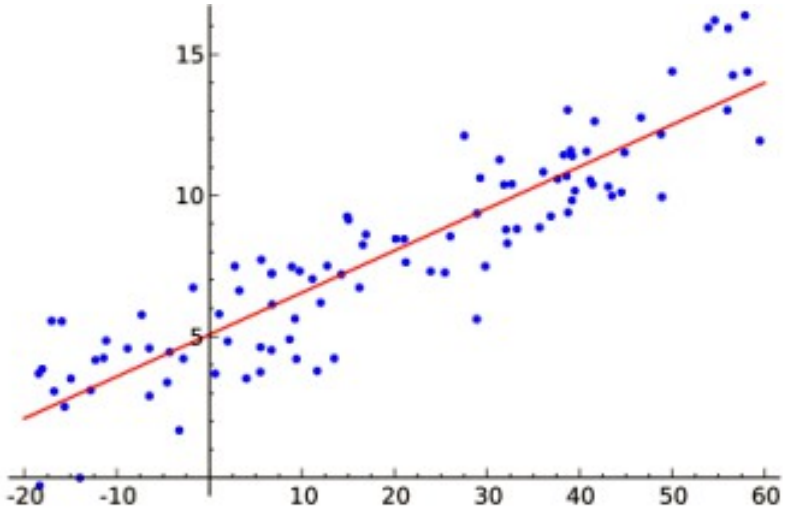

Figure 4.1: solution of LS method

Therefore, the problem to minimize leads to minimize the norm of the residue $||Ax - y||$. To deepen the various mathematical steps, please, refer to [29].
As we will see the in chapter 4, since the data comes from experimental measurements and thy are affected by noise, it is convenient to approximate to least squares. In our problem, we want to minimize the equation $||P1 - R * (P2 - C)||$, where P1 is the vector formed by the points of the first trajectory, while P2 is the vector formed by the related points of the second one.
It is now necessary to parameterize R, and the simplest solution is through the axis-angle representation, better known as *Rodrigues' formula*. We can write $R =$

$I + (s(\vartheta)) * K + (1 - c(\vartheta)) * K^2$, where $s(\vartheta) = sen(\vartheta)$ and $c(\vartheta) = cos(\vartheta)$, which is equivalent to the rotation matrix

$$R = \begin{bmatrix} c(\vartheta) + k_x^2(1 - c(\vartheta)) & k_x k_y(1 - c(\vartheta)) - k_z s(\vartheta) & k_y s(\vartheta) + k_x k_z(1 - c(\vartheta)) \\ k_z s(\vartheta) + k_x k_y(1 - c(\vartheta)) & c(\vartheta) + k_y^2(1 - c(\vartheta)) & -k_x s(\vartheta) + k_y k_z(1 - c(\vartheta)) \\ -k_y s(\vartheta) + k_x k_z(1 - c(\vartheta)) & k_x s(\vartheta) + k_y k_z(1 - c(\vartheta)) & c(\vartheta) + k_z^2(1 - c(\vartheta)) \end{bmatrix} \tag{4.5}$$

Obviously, in the absence of rotation, the matrix is reduced to identity.
The matrix R allows you to rotate a vector in the space, given an axis and an angle of rotation. I is the identity matrix, K is a skew symmetric matrix, in which the unknown components k_1, k_2, k_3 indicate around which axis the trajectory is rotating and how much, representing by the angle ϑ [30].
To minimize the equation we use the Levenberg-Marquardt algorithm. This method is used to solve nonlinear least squares problems. The LM algorithm "can be thought as a combination of steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method" [31], so it is slow but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method. For further clarification about the algorithm, see [32].
The final goal is to find values of ϑ and K that optimizes the initial problem. After finding our unknown quantities solving a single equation, we realized that the values found were far from what we expected, so we decided to face the problem in a different way.
At the beginning, we assume to know of how the trajectory is translated and we want to estimate the rotation matrix. Subsequently, after finding the matrix that best fits the problem, we impose to know of how the trajectory rotates, using the newly calculated matrix and around which axis and then we calculate the translation vector. After computing at the first iteration the rotation matrix and the translation vector, we repeat the procedure until we notice the convergence of the values in a predetermined bound. The problem with this method is that it is only valid for absolute measures.

4.2 Hand-eye calibration method

We are looking for a method that will ensure a solution with measures of a relative nature.
In this section, we consider a system composed of rigidly coupled cameras with non overlapping view, setup that, nowadays, is very common, for example, in the automotive industry. In Figure 4.2, there is an example of a multi-camera systems. It consists of two camera clusters, one on each side of a vehicle. The cameras are attached to the car and can be considered a rigid body [42]. In Figure 4.3, we can see another example of multi-camera system in automotive field, used to reduce

Figure 4.2: Multi-camera system

accidents in working environments. In fact, multi-camera system for forklift trucks facilitates precision manoeuvring.

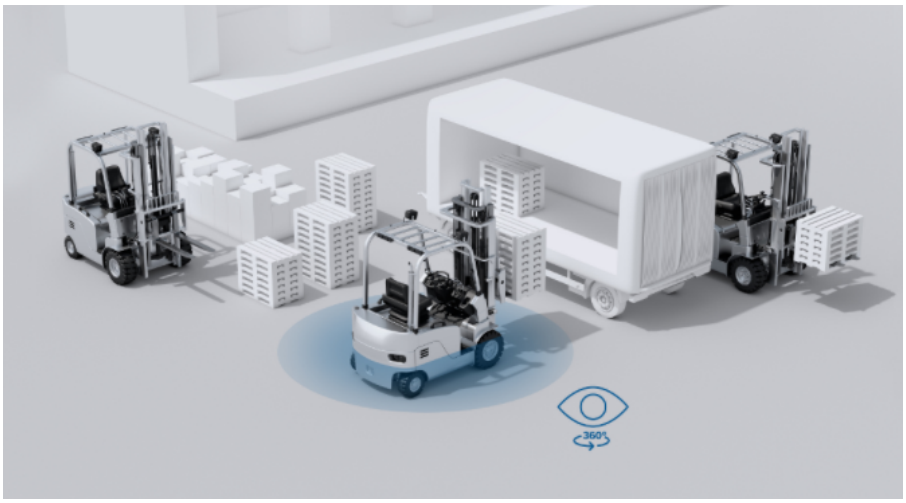

Figure 4.3: Multi-camera system for forklift trucks facilitates precision manoeuvring

At first, considering our model, let us imagine to generate two trajectories, seen in the local camera reference frame, with two cameras, simulating a rigid body. Each camera, assuming that there is not an overlapping view, captures an individual sequence, processed by a structure and motion algorithm. The transformations between the two cameras are estimated using the rigidity constraint of the rig. Let us consider two trajectories in their local reference systems. The change between two Cartesian reference frame is described by a similarity transformation, that is of the form

$$T = \begin{pmatrix} \lambda R & C \\ 0^T & 1 \end{pmatrix} \tag{4.6}$$

where λ is the scale, R is the rotation matrix and C is the translation between the two reference frames.
Through the reading of the trajectories, we can obtain two rotation matrices, that

we call R_{0k} and R_{ik}, in which the subscript *0* represents the master camera and the subscript i represents the client camera, at time k. In the same way, we can get the position vectors, C_{0k} and C_{ik} and consequently the transformation matrices T_{0k} and T_{ik} [33].

$$T_i^k = \begin{pmatrix} R_i^k & C_i^k \\ 0^T & 1 \end{pmatrix} \tag{4.7}$$

Obviously the initial pose of each camera is then given by $R_i^0 = I$ and $C_i^0 = [0\ 0\ 0]^T$.
To switch from one reference system to the other one, we can act in two ways. By first changing system of reference of the master at time k and then by using the transformation [33].

$$T_0^k \Delta T_i \tag{4.8}$$

or, by first applying the transformation and then by changing the system of reference at time k.

$$\Delta T_i T_0^k \tag{4.9}$$

So, at the end, we will have

$$T_0^k \Delta T_i = \Delta T_i T_0^k \tag{4.10}$$

Decomposing the equation 4.10 into one constraint regarding only rotation, it becomes

$$R_0^k \Delta R_i = \Delta R_i R_0^k \tag{4.11}$$

while, linking both orientation and position constraints, the equation 4.10 becomes

$$R_0^k \Delta C_i + C_0^k = \Delta\lambda_i \Delta R_i C_i^k + \Delta C_i \tag{4.12}$$

When the rig rotates and translates, first we consider the rotation through the equation 4.11, and then we consider the position with the equation 4.12.
Replacing, in the equation 4.11, the rotation matrices by quaternions q, we will have

$$q_0^k \Delta q_i = \Delta q_i q_0^k \tag{4.13}$$

or, equivalently

$$(T_{q0^k} - T^*_{q0^k})\Delta q_i = 0 \tag{4.14}$$

where T_q and T_q^* define left and right multiplication with quaternion $q = (w, x, y, z)^T$. Let us derive a linear system of equations with unknows $\Delta q_i = (\Delta w_i\ \Delta x_i\ \Delta y_i\ \Delta z_i)^T$, imposing that $|\Delta q_i| = 1$,

$$\begin{pmatrix} w_0^k - w_i^k & -x_0^k + x_i^k & -y_0^k + y_i^k & -z_0^k + z_i^k \\ x_0^k - x_i^k & w_0^k - w_i^k & -z_0^k - z_i^k & y_0^k + y_i^k \\ y_0^k - y_i^k & z_0^k + z_i^k & w_0^k - w_i^k & -x_0^k - x_i^k \\ z_0^k - z_i^k & -y_0^k - y_i^k & x_0^k + x_i^k & w_0^k - w_i^k \end{pmatrix} \begin{pmatrix} \Delta w_i \\ \Delta x_i \\ \Delta y_i \\ \Delta z_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.15}$$

Now, to solve the problem, it is sufficient to find the eigenvector of A associated with its smallest positive eigenvalue, thus obtaining the quaternion that will form our rotation matrix [34], where

$$A = \sum_{i=1}^{N} (A_i^k)^T * A_i^k \tag{4.16}$$

However, this method is not effective for the presented setup.
This problem is due to the fact that, since there are many sources of perturbations errors associated with camera calibration, the parameters of models robots are not perfect. It follows that the estimation of the hand-eye transformation has errors associated with it and it is important to quantify these errors in order to determine the stability of a given method.
Therefore, in the following, we will introduce another method based on the Iterative Closest Points (ICP) algorithm.

4.3 Iterative Closest Point method

As the method described above does not work properly, we examine a new procedure to estimate the rotation matrix.
We now consider two simple straight trajectories. We push them to the Iterative Closest Point and at the end, we find the estimated rotational matrix ΔR_{i}. So, once the internal rotation ΔR_{i} has been found, we rely again on the method proposed in [33] in order to find the translation vector ΔC_{i}, that can be found by starting from the equation 4.12 and so by solving the following linear system, where $\Delta\lambda_{\mathrm{i}}$ is the scale and it is equal to 1 [33].
In this case, the two trajectories used will not be straight, but they will be two general trajectories, otherwise the algorithm is not observable.

$$\underbrace{\begin{pmatrix} I - R_0^k & \Delta R_i C_i^k \end{pmatrix}}_{B_i^k} \begin{pmatrix} \Delta C_i \\ \Delta\lambda_i \end{pmatrix} = C_0^k \tag{4.17}$$

Chapter 5

Numerical simulation and real world experiments

In this chapter, we deal with the experimental validation of the methods introduced in the chapter 4. After verifying that all the methods work properly, we prove that they are also valid experimentally, testing them first in simulation and then with real data. With regard to simulated data, we used both trajectories derived in simulation and from a dataset. We used a dataset simulating the navigation of the robot. The data is collected in photo-realistic simulation environments with the presence of moving objects, changing light and different weather conditions [43]. The data obtained in simulations are organized in different groups, such as stereo RGB image, depth image, segmentation, optical flow, camera poses and LIDAR point cloud. Among all these, we decided to start from the depth images, collected by both left and right cameras.
Figure 5.1 shows what the left camera sees, while Figure 5.2 shows what the right camera sees.
The dataset contains also two text-files reporting the real trajectory traveled by the two cameras, namely left and right. Each line in the file contains seven elements, of which, the first three elements represent the position along the x-axis, y-axis and z-axis and the remaining four values represent quaternions, which provide a mathematical notation for the representation of orientations and rotations of objects in three dimensions.

5.1 Levenberg-Marquardt method

Given two trajectories, after finding mathematically the rotation matrix and the translation vector which specifies how these are placed between them, we simulate a straight trajectory to verify that our assumptions were confirmed.
We imposed that the true values of the rotation angles were 0° around the x-axis,

Figure 5.1: Room seen from left camera

Figure 5.2: Room seen from right camera

0° around the y-axis and 90° around the z-axis.
By referring to the pseudocode 1, we define:

- the initial trajectory P1 and a hypothetic translation vector and a hypothetic rotation matrix;

- a function *lsqnonlin*, that solves non-linear least squares problems and returns as result the variable $LSlsqR$;

- a function *Rodriguez*, that solves the Rodrigues' formula finding the estimated rotation matrix R_{hat}.

Algorithm 1: Rotation matrix

Data: P1
Hypothetic Data: R, C
Initialization;
$P2 = R * P1 + C$;
$LMlsqR = lsqnonlin(P1, P2, C)$;
$R_{hat} = Rodriguez(LMlsqR)$;
return R_{hat};

Through the algorithm, we found as final angles values 0° around the x-axis, 0° around the y-axis and -90° around the z-axis. As you can see there is an error on the estimation of the angle around the z axis, but this is due to the fact that we were estimating the opposite rotation, then the estimate is correct. In effect, by modifying the algorithm setting, also in this case, we have the desired final result. As regards the displacement, initially we imposed a translation of -0.2335 meters along the x-axis, -0.0890 meters along the y-axis and 0.0081 meters along the z-axis. After running the algorithm reported in Algorithm 2, the estimated values found coincide with the initial ones.

Algorithm 2: Translation vector

Data: P1 and R_{hat}
Hypothetic Data: C
Initialization;
$P2 = R_{hat} * P1 + C$;
$LMlsqC = lsqnonlin(P1, P2, R)$;
$C_{hat} = Rodriguez(LMlsqC)$;
return C_{hat};

The pseudocode Algorithm 2 is very similar to the 1, but there are small differences, in fact there is:

- a function *lsqnonlin*, that solves non-linear least squares problems and returns as result the variable $LSlsq_C$;
- a function *Rodriguez*, that solves the Rodrigues' formula finding the estimated translation vector C_{hat}.

Now, starting from the same values as previously chosen for the angles and the translation, let us add a noise with zero mean value and 0.01 of standard deviation, to each coordinate of each trajectory point.
Also in this case, as in the ideal case, the estimated values are equal to those initially imposed.
Subsequently, in order to stress the algorithm and try it even more, we used also non-straight trajectories.
We imposed again a rotation around the z-axis of 90° and a translation of -0.2335

meters along the x-axis, -0.0890 meters along the y-axis and 0.0081 meters along the z-axis. Also in this case, after executing the code, we found a rotation around the z-axis of -90°, due to the fact that we were estimating the opposite rotation. As regard the translation, the estimated values coincide with those calculated at the beginning.
Here too, we added a noise with zero mean value and 0.01 of standard deviation to each coordinate of each trajectory point and we noticed that the obtained results are as those studied in the previous case.
Now we obtain, from a dataset, a real trajectory and we simulated that this was rigidly attached to another camera.
We applied again the above method and we found the new values of the rotation matrix R and the translation vector C.
In this case, not knowing the initial values, we can not make a comparison between the values imposed and estimated, so we compute the mean value and the standard deviation.
In the same way, we calculate the mean value and the standard deviation for the displacement vector.
In the table 5.1 are shown the mean value and the standard deviation of the rotation angles around x, y and z axes, that we respectively define ϑ, φ and ψ. While, in the table 5.2 there are the mean value and the standard deviation of the translation along the x, y and z axes, that we can define C_x, C_y and C_z. As we can notice,

	mean value [degrees]	standard deviation [degrees]
ϑ	-0.0956	0.0010
φ	0.4035	0.0009
ψ	0.0601	0.0069

Table 5.1: Mean value and standard deviation of the rotation matrix

	mean value [meters]	standard deviation [meters]
C_x	-0.2317	0.0037
C_y	-0.0645	0.0007
C_z	0.0116	0.0006

Table 5.2: Mean value and standard deviation of the translation vector

the standard deviation assumes small values, as we hoped.

5.2 Hand-eye calibration method

In order to test this method, thanks to the Simulink tool, we used a block *rigid body*, that after receiving as input the variables the body mass, the forces acting on it, the inertia matrix and the time, it returns a trajectory.
Thus, we generate two trajectories, seen in the local camera reference frame, with two cameras, simulating a rigid body.
We have arbitrarily imposed the position of two cameras in the body reference system and then we computed the trajectories in the local camera reference system.
In Algorithm 3 is shown the pseudocode of the method that allows to estimate the angles and the displacement, explained in the chapter 4.

Algorithm 3: Computation of the rotation matrix

Data: R_0 and R_i
$q_0 = rotm2quat(R_0)$;
$q_i = rotm2quat(R_i)$;
$T_0 = q_0(:, k)$;
$T_i = q_i(:, k)$;
$A_i = T_0 - T_i$;
$A = A_i^T * A_i$;
$eigenvectors = eig(A)$;
$\Delta R = quat2rotm(eigenvectors)$;
return $eigenvectors$ and ΔR;

As regard the translation, the various steps to be taken to find the final result are shown in the Algorithm 4. Formulae are those mentioned in the chapter 4.

Algorithm 4: Computation of the translation vector

Data: C_0 and C_i
Use a formula to compute B_i;
Use a formula to compute ΔC;
return ΔC;

As we expected from the theory, the method does not respect our requirements. In fact, as we can see in the table 5.3 and in the table 5.4, the estimated values are far from the initial ones. In order to notice even more the difference between the two values, we computed the error as the norm of the difference between estimated and expected value.

$$err_R = |\Delta R - R_0| \tag{5.1}$$

$$err_C = |\Delta C - C_0| \tag{5.2}$$

	initial values [degrees]	estimated values [degrees]
ϑ	-25.0376	180
φ	0	0
ψ	0	180

Table 5.3: Initial values and estimated values of the angles

	initial values [meters]	estimated values [meters]
C_x	0	0.4699
C_y	0	-1.2019
C_z	0	0

Table 5.4: Initial values and estimated values of the translation

As we can see in Figure 5.3 for the translation vector and in Figure 5.4 for the angles of rotation, the fact that the errors are not uniform is due to a large difference between the two considered values.

5.3 Iterative Closest Point method

Since the method described in the previous section do not meet our requirements, we adapted it. Starting from the procedure described in [33], we modified it using the same setup but not the same formulas presented. Therefore, we have decided to use the Iterative Closest Point method, as it is able to achieve as a result the transformation, i.e. a combination of rotation and translation, between a reference and a source, and that is in our case the master camera and the client camera. The ICP method is one of the most used models when you have as initial guess a rigid transformation.
In order to verify the proper functioning of the algorithm, we examine the same setup considered in the previous section. We look at a rigid body, on which no external forces act, thus it goes straight and it generates straight lines. Since the problem examines the case with two cameras, for simplicity, we consider that the trajectory of the center of mass of the body coincides with the trajectory of the master camera.
We impose that the slave camera is rotated 45° along the x-axis, 30° along the y-axis and 90° along the z-axis and is translated of 2 meters along the x-axis and 1 meter along the y-axis. In this first case then, we consider two simple trajectories. The ICP algorithm, as shown in Algorithm 5, solves our problem:

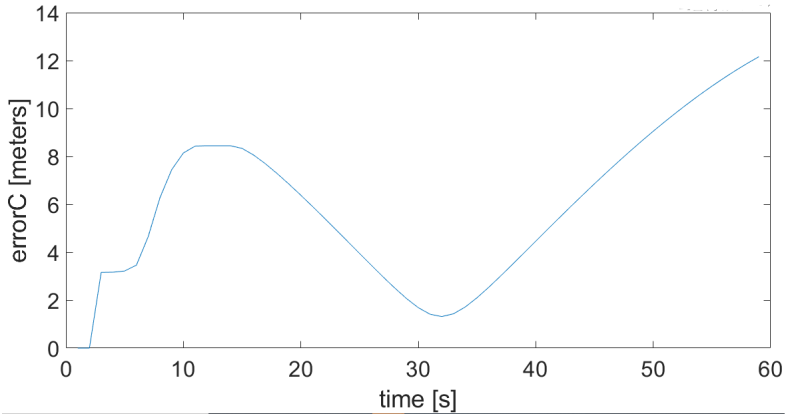

Figure 5.3: Norm of the translation vector

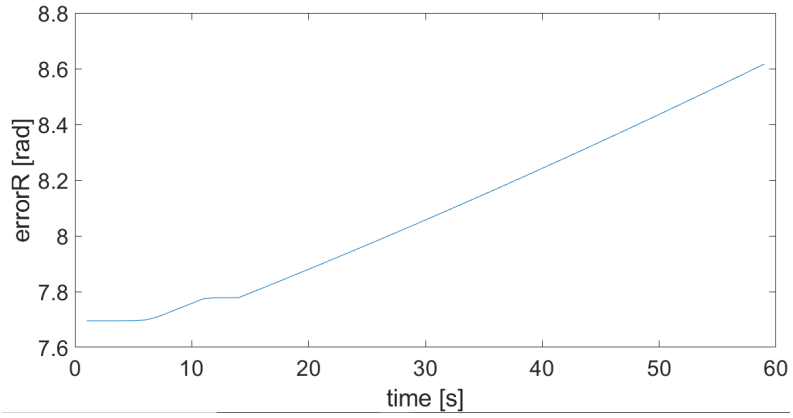

Figure 5.4: Norm of the rotation matrix

- we define P_hat, that is where the master camera starts;
- a function *pointCloud* computes the lines followed by the master and the client cameras;
- a function *pcregistericp* registers two point clouds using ICP algorithm;
- compute the new rotation matrix ΔR.

In the ideal case, the estimated values are equal to the initial ones.
On the other hand, by calculating the vector, there is a significant difference between the estimated values and the initial ones, as we can see in the table 5.5.
Thus, we modified the algorithm using generic trajectories rather than straight trajectories, as shown in Algorithm 6.
In this way, the estimated values of the translation vector are equal to the initial ones.

Algorithm 5: Computation of the rotation matrix

Data: P_{hat}
Hypothetic Data: R
$P0 = Inizialize_straight_traj()$;
$P1 = P0 + R * P_hat$;
$line0 = pointCloud(P0)$;
$line1 = pointCloud(P1)$;
$tform = pcregistericp(P0, P1)$;
$\Delta R = tform.Rotation$;
return ΔR

Algorithm 6: Computation of the translation vector

Data: P_{hat}
Hypothetic Data: R
$P0 = randn()$;
$P1 = P0 + R * P_hat$;
$traj0 = pointCloud(P0)$;
$traj1 = pointCloud(P1)$;
Use a formula to compute B_i;
Use a formula to compute ΔC;
return ΔC;

	initial values [meters]	estimated values [meters]
C_x	2	-0.0589
C_y	1	0.025
C_z	0	0.120

Table 5.5: Initial values and estimated values of the translation with the ICP

Then, we try to study the trajectories to modify of the uncertainty, seeing how estimates vary as the noise changes.
So, we add a noise to every point of the trajectory, considering a range within which the noise varies, for example between 0 and 0.1 with a step of 0.01, in order to have at the end a series of different trajectories with different amounts of noise and we repeated the above calculations. Obviously, in this situation where we are not in the ideal case, thus the values found will not be equal to the initial ones, but they are very close. In fact, in the table 5.6, we can see the values of the angles, while in the table 5.7 we can see the values of the translation.

	initial values [degrees]	estimated values [degrees]
ϑ	45	44.13
φ	30	29.95
ψ	90	89.08

Table 5.6: Initial values and estimated values of the angles with the ICP

The veracity of the results is also shown by calculating the error as the norm of the difference between estimated and expected value.

$$err_R = |\Delta R - R_0| \tag{5.3}$$

$$err_C = |\Delta C - C_0| \tag{5.4}$$

As displayed in Figure 5.5 for the angles and in Figure 5.6 for the displacement, the error settles then the two measures acquire very similar values. Moreover, having considered a trajectory in which noise varies over time, we can show boxplots, that highlight the various changes. For semplicity, in Figure 5.7, are shown the boxplots of the estimated angles of a single noise level, rather than having as many boxplots as there are noise levels.
The red line indicates the median, and the bottom and top edges of the box indicate the 25^{th} and 75^{th} percentiles, respectively.

	initial values [meters]	estimated values [meters]
C_x	2	1.75
C_y	1	1.02
C_z	0	0

Table 5.7: Initial values and estimated values of the translation with the ICP

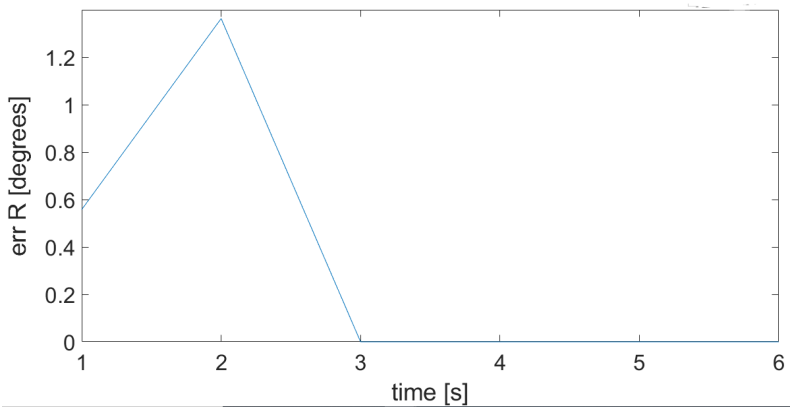

Figure 5.5: Norm of the rotation matrix

In Figure 5.8, there is the graph that represents the three coordinates considering, also in this case, a single noise level.
The more flat the boxes, the more correct the result, because it means that there is not much variance.
After verifying that the algorithm works, we used a real trajectory, caught by a camera. For this work, the RealSense T265 was chosen.

5.4 Real world measurements

To the trajectory generated from the camera, we rigidly attached another camera, thus obtaining a final trajectory. Let us imagine a rigid system in which there are two cameras, placed as shown in Figure 5.9, that move simultaneously. The camera on the right of the rigid body is the client, the other one is the master.
The system, moving, registers a trajectory, that is saved in some rosbags. In our experiment, we have two kind of bags, one representing a straight line and the other one representing a general trajectory. These acquisitions are done under controlled conditions to assess the estimation error.
Before reviewing the code, we need to make sure that the acquisitions of both cameras are synchronized. In order to do this, it is necessary to know the time stamp of

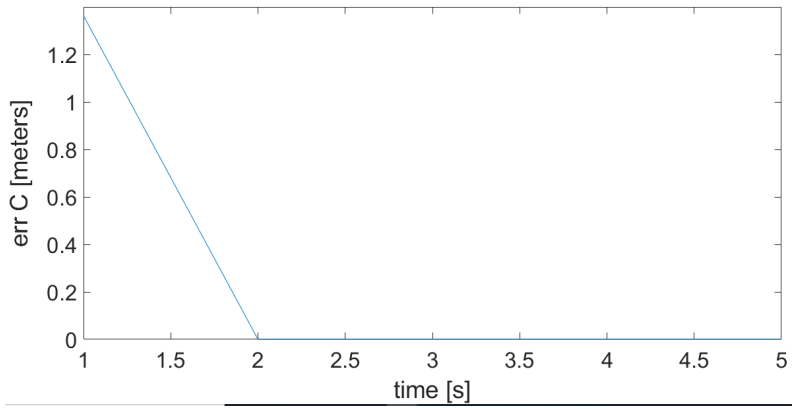

Figure 5.6: Norm of the translation vector

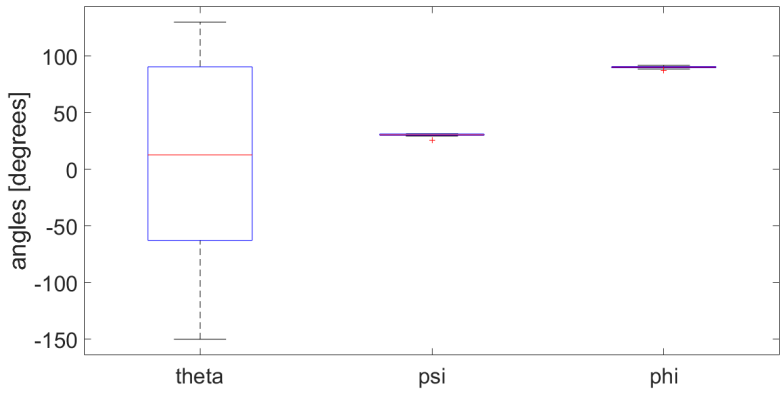

Figure 5.7: Boxplot of the estimated angles

the master camera and the client camera and make sure that the i-th element of one corresponds to the i-th element of the other one. Since, in our case, the acquisition is not synchronous, we will never have the same time stamp for the two cameras. Thus, it is indispensable associate each element of the client to the master, since the latter is composed of fewer elements. We impose a threshold and we affirm that all the data that fit into its window, is as if they were contemporary. The difference between the two acquisitions is really minimal, in fact we are able to measure it only if we consider the nanoseconds. In Algorithm 7 are shown the various steps just described. In order to estimate the rotation matrix, we utilize the straight line, while to estimate the position vector, we use the general trajectory and we proceed according to the ICP algorithm and we repeat the procedures explained in the previous section.
In table 5.8 the imposed and the estimated values of the angles are shown, while in the table 5.9 there are the same values, but regarding the translation. As expected, the values are similar to the imposed ones. Also in this case, we find the

Algorithm 7: Synchronization of the acquisitions

```
1 Impose a threshold;
2 Start from msgStructs_M;
3 Start from msgStructs_C;
4 return the difference of nanoseconds;
5 while the difference is greater or equal to the threshold do
6     if the length of the master has been exceeded then
7         return;
8     else
9         Compute the new difference;
```

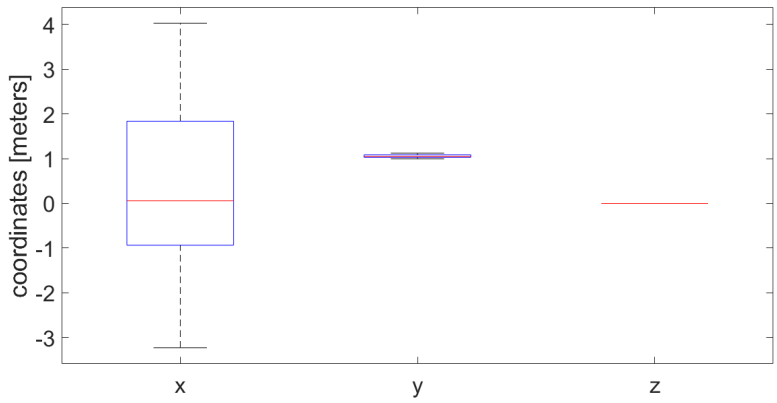

Figure 5.8: Boxplot of the estimated distance

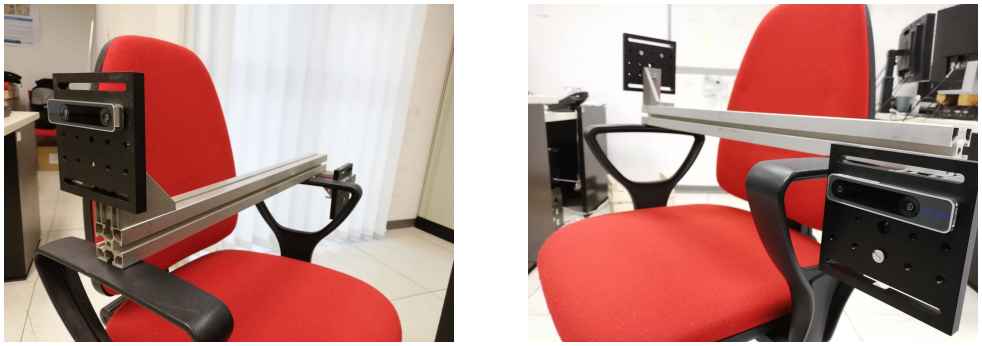

Figure 5.9: Rigid system

negative sign only because we consider the opposite direction of rotation.
At this point, to test even more the algorithm, always starting from the two cameras, we generated again two type of trajectories, one straight and one general. On these paths, we made a series of acquisitions, to which we have applied the above mentioned procedures.
At first, our initial measurements of the angles of rotation and the distance between the two cameras were measured by hand, so inaccurately and with systematic errors of measures. In this way, we do not know the real initial value. Therefore, the ultimate goal is not to find the error of estimation, but the variance and the standard deviation.
In the table 5.10 the values of the angles are shown, while in the table 5.11 the values of the displacement are shown. As we hoped, the standard deviation should converge to a low value, which means that our value is very close to the real value. Subsequently, we were able to do controlled acquisitions and to compare the initial and final measurements, as shown in the table 5.12 for the angles and in the table 5.13 for the displacement. As we can see the estimated values are very close to those calculated.

	initial values [degrees]	estimated values [degrees]
ϑ	180	-179.6
φ	0	2.90
ψ	90	101.19

Table 5.8: Initial values and estimated values of the angles with the real acquisitions

	initial values [meters]	estimated values [meters]
C_x	-0.66	-1.16
C_y	0.045	0.049
C_z	-0.21	0.41

Table 5.9: Initial values and estimated values of the translation with the real acquisitions

Moreover, in Figure 5.10 the boxplots of the estimated angles are shown, while in Figure 5.11 there are the boxplots of the displacement, and since the boxes are flat, we can notice that the estimates are very good.

	mean value [degrees]	standard deviation [degrees]
ϑ	179.3388	0.3460
φ	-0.1756	0.1459
ψ	86.9271	0.9888

Table 5.10: Mean value and standard deviation of the angles with the real acquisitions

	mean value [meters]	standard deviation [meters]
C_x	-0.0839	0.5251
C_y	0.0616	0.5537
C_z	-0.0350	0.3311

Table 5.11: Mean value and standard deviation of the translation with the real acquisitions

	initial values [degrees]	estimated values [degrees]
ϑ	180	179.3388
φ	0	-0.175
ψ	90	88.93

Table 5.12: Initial values and estimated values of the angles with the real acquisitions

	initial values [meters]	estimated values [meters]
C_x	-0.66	-0.083
C_y	0.09	0.072
C_z	-0.21	-0.135

Table 5.13: Initial values and estimated values of the translation with the real acquisitions

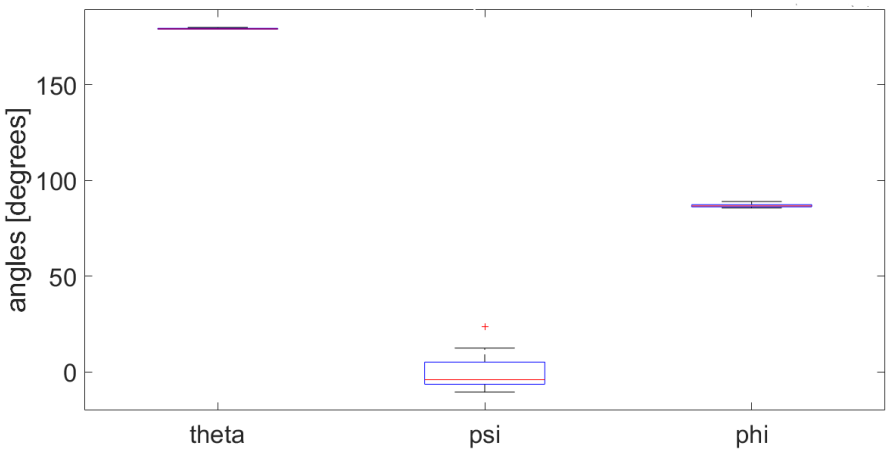

Figure 5.10: Boxplots of the estimated angles

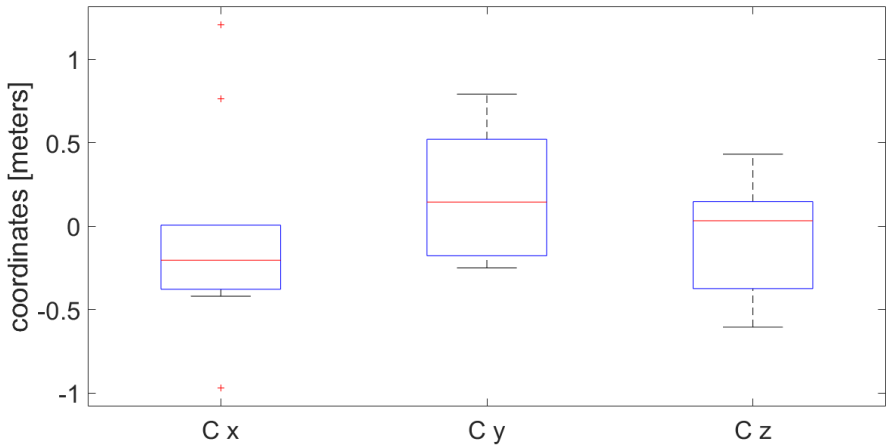

Figure 5.11: Boxplots of the estimated distance

Chapter 6

Conclusions

The aim of the thesis is to propose new techniques in order to estimate the pose of a multi-camera systems. In our case, we used only two cameras, that are able to capture a sequence of 3D point clouds. Thanks to the collected data, each camera estimate a trajectory, through SLAM algorithms. Then, starting from the trajectory, we used extrinsic calibration methods in order to compute the relative pose between the two cameras.
The initial idea was to use a dataset for robot navigation tasks, whose data collected in simulation by two cameras produced a trajectory. However, we have notice that SLAM algorithms did not work well in more complex scenarios, where for example there were challenging viewpoints or diverse motion patterns. In fact, these situations are difficult to achieve by using physical data collection platforms. Thus, we studied a series of extrinsic calibration methods in order to achieve our goal.
The setup is based on two cameras rigidly coupled and estimating the trajectory accomplished by these, we have been able to know the constraint that exists between them, that is how much they are shifted and rotated between them.
The choice of using SLAM algorithms compared to other techniques, for example the *Global Positioning System or GPS*, has been dictated by specifications related to measurement uncertainty and feasibility. In fact, the GPS is accurate compared to a global scale, so the accuracy decreases equated to a room or a small area. Moreover, it is a satellite-based system, suffering from physical limitations. SLAM algorithms, instead, use as much data as possible to create the surrounding environment map.
In order to satisfy the accuracy requirements, we developed a series of methods. The first one, the Levenberg-Marquardt, does not fully meet our specifications, because it is valid only for absolute measures. Instead, the procedure based on the hand-eye calibration works partially, i.e. it perfectly estimates the rotation, but is not able to estimate the translation as well. At the end the last procedure, that is the combination between the last method and the Iterative Closest Point, converges to the optimal solution.

After testing the simulation algorithms, we wanted to use real data and we had to use a robot to record the trajectory taken and collect the data. The actual robot, shown in Figure 6.1, is the 4WD Four wheel steering robot with suspensions for autonomous navigation and outdoor applications, designed and developed by *robodyne srl*. Unfortunately, for timing problems and other situations, we have

Figure 6.1: 4WD Four wheel steering robot with suspensions for autonomous navigation and outdoor applications

never been able to use it. That is why, we adapted and used a more domestic system, using a chair with two cameras to reproduce a trajectory.
What we set out to do, has been completed, i.e. to estimate the relative pose of the two cameras.
However, in the future, one might consider continuing the work studying methods to improve accuracy and precision. A solution could be to consider a more controlled environment and initially less subject to noise or making the environment more infrastructural.

Bibliography

[1] Come i robot trasformeranno l'agricoltura,ht-apps.eu, March 2020.

[2] *Alessandro Zorer*, I farmbot e la diffusione dei robt in agricoltura, makerfairerome.eu, April 2020.

[3] *Gian Basilo Nieddu*, Farmdroid: il robot agricolo alimentato dal sole, vaielettrico.it, July 2020.

[4] Mobile Robots in Industry 4.0: automation and flexibility, robotnik.eu, January 2021.

[5] *Sarah Moore*, Applications of Machine Vision in Robotics, www.azorobotics.com, December 2019.

[6] *Wang Qi, Fu Li and Liu Zhenzhong*, Review on Camera Calibration, Control and Decision Conference (CCDC), 2010 Chinese, June 2010.

[7] *Wang Qi, Fu Li and Liu Zhenzhong*, Instrument Calibration for the 21st Century, MSPS 57th Annual Meeting, St. Cloud, MN, January 2009.

[8] *Renbo Xiaa, Maobang Hua, Jibin Zhaoa, Songlin Chena, Yueling Chena and ShengPeng Fua*, Global calibration of non-overlapping cameras: State of the art, Optik - International Journal for Light and Electron Optics, December 2017.

[9] *Ram Krishan Kumar, Adrian Ilie, Jan-Michael Frahm and Marc Pollefeys*, Simple Calibration of Non-overlapping Cameras with a Mirror, 2008 IEEE Conference on Computer Vision and Pattern Recognition, June 2008.

[10] *Pierre Lébraly, Eric Royer, Omar Ait-Aider and Michel Dhome*, Calibration of non-Overlapping Cameras-Application to Vision-Based Robotics, Proceedings of the British Machine Vision Conference, January 2010.

[11] *Branislav Micusik*, Relative pose problem for non-overlapping surveillance cameras with known gravity vector, Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, July 2011.

[12] *Peter Sturm and Thomas Bonfort*, How to Compute the Pose of an Object without a Direct View?, Asian Conference on Computer Vision, January 2006.

[13] *Qianzhe Liu, Zhen Liu, Junhua Sun and Guangjun Zhang*, Global calibration method of multi-sensor vision system using skew laser lines, Chinese Journal of Mechanical Engineering 25, March 2012.

[14] *Giulio Figari*, Rubrica: Image Processing – Pt2, gpemmocap.wordpress.com, March 2017.

[15] *Martin Ingvaldsen*, Understanding the importance of 3D hand-eye calibration, blog.zivid.com, November 2019.

[16] Cos'è un sensore e a cosa serve?, dewesoft.com, March 2020.

[17] *Jackson Morgan*, Light Sensors: Units, Uses, and How They Work, blog.endaq.com, October 2018.

[18] SENSORI DI TEMPERATURA, luchsinger.it.

[19] *Gianluigi Torchiani*, Sensori ottici: cosa sono, come funzionano, tipologie e ambiti applicativi, internet4things.it, January 2020.

[20] Come calibrare un sensore per misure accurate, inquinamenti-italia.com.

[21] *Faraz M. Mirzaei*, Extrinsic and Intrinsic Sensor Calibration, University of Minnesota, December 2013.

[22] *Jonathan Brookshire and Seth Teller*, Extrinsic Calibration from Per-Sensor Egomotion, MIT Computer Science and Artificial Intelligence Laboratory.

[23] *Shuai Dong, Fujun Yang and Xinxing Shao*, Extrinsic Calibration of a non-overlapping camera network based on close-raneg photogrammetry, Applied Optics, August 2016.

[24] *Qilong Zhang and Robert Pless*, Extrinsic Calibration of a Camera and Laser Range Finder, Intelligent Robots and Systems, Intelligent Robots and Systems, January 2004.

[25] *Subodh Mishra, Gaurav Pandey and Srikanth Saripalli*, Extrinsic Calibration of a 3D-LIDAR and a Camera, Intelligent Vehicles Symposium, May 2020.

[26] *Junhao Xiao, Chenghao Shi, Kaihong Huang and Qinghua Yu*, Extrinsic Calibration and Odometry for Camera-LIDAR Systems, Modeling the Disaster Environment Using Intelligent Mobile Robots, September 2019.

[27] *Jun Zhang, Prarinya Siritanawan, Yufeng Yue and Chule Yang*, A Two-step Method for Extrinsic Calibration betvween a Sparse 3D LIDAR and a Thermal Camera, 15th International Conference on Control, Automation, Robotics and Vision, November 2018.

[28] *Andrea Onofri e Dario Sacco*, Metodologia sperimentale per le scienze agrarie, Capitolo 11: I minimi quadrati, October 2010.

[29] *Andrea Onofri e Dario Sacco*, Least squares method,CNR.

[30] *Paolo medici*, Parametrizzazione Asse-Angolo, November 2017.

[31] *Manolis I. A. Lourakis*, A Brief Description of the Levenberg-Marquardt Algorithm Implemened, Institute of Computer Science, Foundation for Research and Technology, February 2005.

[32] *Henri P. Gavin*, The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems, Department of Civil and Environmental Engineering Duke University, September 2020.

[33] *Sandro Esquivel, Felix Woelk, and Reinhard Koch*, Calibration of a Multi-camera Rig from Non-overlapping Views, Pattern Recognition, Christian-Albrechts-University, 24118 Kiel, Germany.

[34] *Radu Horaud and Fadi Dornaika*, Hand-eye Calibration, HAL archives-ouvertes, May 2011.

[35] *Prochazkova Jana and Martisek Dalibor*, NOTES ON ITERATIVE CLOSEST POINT ALGORITHM,17^th Conference on Applied Mathematics APLIMAT, Slovak University of Technology in Bratislava, Faculty of Mechanical Engineering, April 2018.

[36] *Nabil Madali*, Structure from Motion, towardsdatascience.com, June 2020.

[37] *Abdou Shalaby, Mohammed Elmogy and Ahmed Abo El-Fetouh*, Algorithms and Applications of Structure from Motion (SFM): A Survey, International Journal of Computer and Information Technology, November 2017.

[38] *JOHAN FREDRIKSSON*, ROBUST ROTATION AND TRANSLATION ESTIMATION IN STRUCTURE FROM MOTION, Lund University, Faculty of Engineering, Centre for Mathematical Sciences, 2016.

[39] *Abdou Shalaby, Mohammed Elmogy and Ahmed Abo El-Fetouh*, Algorithms and Applications of Structure from Motion (SFM): A Survey, International Journal of Computer and Information Technology, November 2017.

[40] *Jana Procházková and Dalibor Martišek*, Notes on Iterative Closest Point Algorithm, Faculty of Engineering, 17th Conference on Applied Mathematics APLIMAT, April 2018.

[41] *Ying He, Bin Liang, Jun Yang, Shunzhi Li and Jin He*, An Iterative Closest Points Algorithm for Registration of 3D Laser Scanner Point Clouds with Geometric Features, Sensors, August 2017.

[42] *Brian Clipp, Jae-Hak Kim, Jan-Michael Frahm, Marc Pollefeys and Richard Hartley*, Robust 6DOF Motion Estimation for Non-Overlapping, Multi-Camera Systems, 9th IEEE Workshop on Applications of Computer Vision, January 2008.

[43] *Wang, Zhu, Wang, Hu, Qiu, Wang, Hu, Kapoor and Scherer*, TartanAir: A Dataset to Push the Limits of Visual SLAM, Cornell University, August 2020.

[44] *Wolfram Burgard, Cyrill Stachniss, Kai Arras and Maren Bennewitz*, Introduction to Mobile Robotics - SLAM: Simultaneous Localization And Mapping, University of Freiburg.

[45] *Norlida Mohamad Yatim and Norlinda Buniyamin*, Particle filter in simultaneous localization and mapping (SLAM) using differential drive mobile robot, Jurnal Teknologi, December 2015.

[46] *Sebastian Thrun*, Simultaneous Localization and Mapping, Springer Tracts in Advanced Robotics.

[47] What is SLAM?- What is SLAM and how does it work?, GeoSLAM.com.

[48] What Is SLAM? 3 things you need to know, mathworks.com.

[49] *Tim Field, Jeremy Leibs, James Bowman, Dirk Thomas*, rosbag, ROS.org.

[50] *Michele Nasi*, Intel presenta la videocamera di tracciamento RealSense T265: cos'è e come funziona, Soluzioni Tecnologiche, January 2019.
[51] *Davide Scaramuzza and Zichao Zhang*, Visual-Inertial Odometry of Aerial Robots, arXiv.org, June 2019.
[52] *Fulvio Barbato*, Intel presenta Intel RealSense T265, spaziotech.it, January 2019.
[53] *Adnan Ademovic*, An Introduction to Robot Operating System: The Ultimate Robot Application Framework, toptal.com, January 2016.
[54] *Yahya Tawil*, An Introduction to Robot Operating System (ROS), allaboutcircuits.com, June 2017.
[55] About ROS, ros.org.
[56] *YoonSeok Pyo, HanCheol Cho, RyuWoon Jung and TaeHoon Lim*, ROS Robot Programming - A Handbook Written by TurtleBot3 Developers, December 2017.