Politecnico of Torino

Department of Control and Computer Engineering

Simultaneous Localization and Mapping for autonomous vehicles

MASTER'S DEGREE IN:

Mechatronics Engineering

PRESENTS :

Antonio Marangi

TUTOR

Professor Eros Pasero (Politecnico of Torino)

Professor Marina Mondin (California State University of Los Angeles)

Professor Marco Lovera (Politecnico of Milano)

2019

Abstract

Simultaneous Localization and Mapping(SLAM) has been considered by literature the holy grail of the robotics because of many aspects, first above all the enhanced ability of generic unmanned vehicles to move in an unknown and unstructured environment. Anyway, SLAM is a chicken-or-egg problem since for localization a map is needed and for mapping the information about the precise position is required, hence the two aspects cannot be approached independently, indeed their reciprocal relationship determines the real core of the matter. The work aims at describing what SLAM is and how it can be implemented from a theoretical point of view. A revision of the literature and research of actual technologies has been carried on. Last but not least an application of this approach has been experimented on an example of service robot acted at escorting and helping airport travelers under the supervision of California State University of Los Angeles in collaboration with the San Diego Airport.

Executive Summary

The purpose of the paper is to provide an overview of the literature about Simultaneous Localization and Mapping matter, its theoretical foundaments and the description on how it can be implemented on a service robot.

Simultaneous Localization and Mapping (SLAM) is the problem of localizing a vehicle in an unknown and unstructured environment while mapping it. It is referred to be a chicken-or-egg problem since the two phases, localization and mapping, cannot be tackled independently since in order to resolve one of them the other is needed. SLAM exploits sensors to perceive reality surroundings the system and creates a map in order to make successive localization faster and more reliable. Example of used sensors are cameras, monocular or stereo, or ToF cameras and lidars. The first exploits only visual information gathered from the lenses, the last uses lasers to scan in three or two dimensions the environment, hence exploiting the time-of-flight data, while ToF cameras are hybrids between the two.

In robotics application like service robots the lack of information about the mission environment create the need of a vehicle able to perceive the reality in real time, which cannot rely on a-priori information about it. This is why approaches like SLAM seems to be so appealing in this field.

As en example of implementation of the SLAM system in the field of service robotics an escorting robot prototype has been built. The robot was conceived under the supervision of California State University of Los Angeles in collaboration with the San Diego Airport. Its aim was to assist airport travelers in the airport, providing services like maps, information and navigation system. A costumer would arrive at the airport and ask the robot to show him the way to check-in desks, gates or restaurants.

The main author contribution to the project has been the development of SLAM design and integration in the existing vehicle. The software, entirely developed in C++ runs on an Ubuntu environment installed on the NVIDIA development board, JETSON TX2. The software is able to take images from the stereocamera, and from these, create the map of the environment in first place. After map was generated the software would localize the robot and send commands to the motors in order to complete the prescribed mission by reaching the final destination decided by the user, who interfaced himself with the autonomous robot by a tablet application.

Contents

1 Introduction to service robotics **1**

1.1 Example of service robots 2

1.1.1 Savioke[1] 2

1.1.2 Airstar 3

1.1.3 Aethon[2] 4

1.2 Robot perception of reality 4

1.2.1 Perception of surrounding and localization methods 5

1.2.2 Interaction with environment 7

2 Simultaneous Localization and Mapping techniques **10**

2.1 Visual SLAM 10

2.2 Visual SLAM sensors 11

2.2.1 Camera preparation for SLAM 18

2.2.2 Image processing for SLAM 22

2.3 Visual SLAM methods and implementations 28

2.3.1 Feature extraction algorithms 31

2.3.2 Keyframes selection and tracking 44

3 Innotech autonomous vehicle **50**

3.1 Schematics 51

3.2 Datasheets 52

3.2.1 ZED stereocamera by Stereolabs 52

3.2.2 Jetson TX2 53

3.2.3 F28069-LAUNCHXL by Texas Instruments 54
3.3 Adopted communication protocols 56
3.3.1 CAN bus 56
3.3.2 MQTT 59
3.4 Shape design 61
3.5 User Interface (UI) 62

4 Robot Localization and Mapping **65**
4.1 Implemented SLAM system 65
4.1.1 Virtual/Real CAN bus 67
4.1.2 Mapping 67
4.1.3 Navigation 69
4.2 Testing 74

5 Conclusions and future lines **78**

References **85**

List of Figures

1.1 Savioke Relay . . . 3
1.2 LG Airstar . . . 3
1.3 Aethon . . . 4
1.4 GPS Trilateration Mechanism . . . 6
1.5 Example of an output map from LiDAR inspection . . . 7
1.6 Sample images of end effectors from Robotiq®[3] . . . 8
1.7 Uncanny valley[4] . . . 9
1.8 Effect of movement on Uncanny valley plot . . . 9

2.1 Example of MonoCamera: Firefly ® S [5] . . . 12
2.2 Distance calculation of known dimensions object . . . 13
2.3 Example of ToF Camera: Basler ToF Camera[6] . . . 13
2.4 3D time-of-flight camera operation (Image taken from Texas Instrument[7]) 14
2.5 Example of Stereocamera: ZED® camera[8] . . . 15
2.6 Simplified Stereo Vision System[9] . . . 15
2.7 Typical Stereo Vision System[9] . . . 16
2.8 Incorrect triangulation example . . . 17
2.9 Stereocamera misalignment error sources[10] . . . 18
2.10 Radial distortion . . . 20
2.11 Decentering Distortion . . . 21
2.12 Effect of tangential distortion . . . 22
2.13 The epipolar constraint . . . 23
2.14 Epipolar geometry . . . 24

2.15 Rectification of an image stereopair 27
2.16 (a)SLAM/SFM as Markov random field. (b) and (c) Inference propagation in filter and keyframe approach 29
2.17 Feature matching 32
2.18 Bresenham circle 33
2.19 Image without points suppression 34
2.20 Image with points suppression 34
2.21 Harris detector output image 36
2.22 Example of successive pyramidal images 37
2.23 Example of successive application of Gaussian blur operator 38
2.24 DOG on the left and Laplacian operator on the right 38
2.25 Integral images area of interest 40
2.26 Haar-wavelet responses based orientation extraction 40
2.27 42
2.28 Pyramid levels of same image 43
2.29 Orientation of interest point patch 43
2.30 ORB extraction and matching 44
2.31 Perspective-3-points problem 45
2.32 47
2.33 Tree-like structure binary vocabulary 48
2.34 Qualitative flow chart of map building procedure 49

3.1 Side view of the prototype 50
3.2 System Architecture 51
3.3 Front view of the prototype 52
3.4 Jetson TX2 schematics 54
3.5 54
3.6 Encoder based control block 55
3.7 Differential aiding 56
3.8 Differential signal example 57

3.9 CAN bus message data structure 58
3.10 CAN data bitmask 59
3.11 mqtt message structure 60
3.12 Possible solutions for robot design 62
3.13 Other solutions for robot design 62
3.14 User interface homepage 63
3.15 Airport map 63
3.16 Interest points list 64
3.17 Language selection 64

4.1 Ramification of the workflow 66
4.2 Manual control flowchart 68
4.3 isnotRange function 71
4.4 Life cycle flow chart 73
4.5 Straight path on the left and L-shaped on the right 74
4.6 Sequence 1 trajectory of the L-shaped department path 75
4.7 Sequence 2 trajectory of the L-shaped department path 75
4.8 Results of survey 77

List of Tables

2.1 Sensitivity to depth precision for different misalignment[10] 19

2.2 Harris detector eigenvalues criteria 36

3.1 Video specifications 53

3.2 Depth specifications 53

3.3 Motion specifications 53

3.4 Sensor specifications 53

3.5 Power specifications 53

3.6 JetsonTX2 54

4.1 Tests Results 76

1 Introduction to service robotics

The term Service Robotics is used to refer to that category of autonomous or automatic unmanned machines able to perform tasks that can be more or less repetitive depending on the kind of application. Service robots are nowadays used in each field of the human life, from the domestic environment to the industrial one. Their increasing capability to effectively interact with clients or workers are making them suitable to be applied in wider and wider range of operations. Nowadays robot are gaining importance and application in those works characterized by dull and repetitive tasks even taking into account the challenges that an unstructured and dynamical environment presents, nonetheless robots are increasingly exploited in more complex environment when they are able to overcome the intrinsic limitations of reality perception. Automated machines that perform repetitive tasks in not a novelty, it is enough to think to the Ford Motor Company, which in 1913 introduced a car production line which is considered one of the pioneer types of automation in the manufacturing industry[11]. With the passing of the years automation in industry became an essential feature in market competitive industries, this approach makes possible an increased production in reduced time, less errors and more repeatability and accuracy[12]. What makes service robotics a completely different paradigm from automatic robots is the ability of working in unknown,changing and crowded environment, the kind of services provided and the complexity of the system.

The way service robots are made suitable for these complex tasks comprehends different characteristics[13] :

1 Sensors which perceive the environment and the own state.

2 Modelling of the sensorial information in order to represent the ascertainable environment as well as the state of the robot.

3 Integration of significant local information to a global and consistent description.

4 Methods to exploit the gathered knowledge to accomplish tasks of different kinds

1.1 Example of service robots

To the aim of introducing the work done some examples of service robotics will be examined in order to understand the actual state of the art of the leader technologies and solutions. Leaving apart the "less-challenging" solutions such as: automatic vacuum dust machines, essential domotic applications, etcetera. this section will briefly describe some exmaple of indoor autonomous robots able to perform various tasks made challenging by the dynamical unstructured environment they operate.

1.1.1 Savioke[1]

Savioke is an autonomous wheeled platform designed for use in the hospitality and service industries. It is meant to operate as butler mainly in hotel facilities, it is able to charge a payload inside a secured trunk and navigate through the environment in order to complete the delivery.

The robot is approximately 1 meter tall, weighs less than 50 Kg, has a carrying capacity of 57 L, and is designed to travel at a human walking pace. It travels independently between floors via the hotel elevator[14].

Figure 1.1: Savioke Relay

The robot interact with people through a user interface implemented on a small tablet thought to be minimal and easy to use. Moreover it is able to navigate and localize itself via sensors such as LiDar and proximity sensors.

1.1.2 Airstar

Airstar is a robot that can help airline passengers navigate the airport. It is fully autonomous and able to understand if obstacles hinder its motion and avoid them. It provides also general information about the airport, congestion status, shops, etcetera. It communicates with users via one big screens positioned on the front and rear central body[15].

Figure 1.2: LG Airstar

It is currently deployed for commercial use at Incheon airport in Seoul, South Korea.

1.1.3 Aethon[2]

Aethon is an example of heavy weights carrier autonomous robot. It is meant to move payloads up to 700 Kg with a speed of 76 cm/s.

Figure 1.3: Aethon

Its development is aimed to free up staff to focus on patient care, and improve service levels. Hospitality environments can gain improvements in efficiency as was well as in the guest experience.

1.2 Robot perception of reality

As previously pointed out the main characteristic of modern service robot applications is the capability of the machines to operate in unpredictable and crowded environment. In industrial automated production deployment robots like mechanical arms or automated carriers can be made aware of the environment a priori, this means that the assumption of no unforeseen events is made, with the only exception of really hazardous situations, and the robot will move inside a always known and unchangeable space. Moreover, aiding the robot with infrastructure changes is simpler in this kind of applications, for example Landmark-based Navigation is convenient in the scenario of a stock that can precisely calculate the increase in efficiency the autonomous vehicle can import, calculate revenues and operate investments in infrastructure according

to that. To make an example of what stated Autonomous ground vehicles (AGV) are widely used in stocks by different companies, among all Amazon is a leader in the sector and on 2018 Deutsche Bank published a market research about Amazon transportation[16] pointing out the introduction of autonomous carriers in stocks cut expenses of about 20% resulting in $22 million in cost savings. This reduce in costs came from the less involved employers and in reduced fixed costs such as less air condition and lights. With these kind of advantages become easy to adapt the environment to a robot but when it comes to machines that operate in contact with people or clients the paradigm is different, the robot must be able to adapt and understand the environment just like a human can do. What is clear is that autonomous robots need sensor of any kind to grasp information from the surrounding and with those gathered it has to take decisions. In the following sections 1.2.1 and 1.2.2 the methods to perceive the environment and the ways the machine interact with it will be described.

1.2.1 Perception of surrounding and localization methods

Localization in an unknown environment is something not novel in today world, in fact Global Positioning System (GPS) satellites were launched in 1978 and became fully operational in 1993 [17]. GPS provides a global position information through the principle of Trilateration and it is considered the De Facto standard in automotive localization sector.

This kind of technology allows a precision of up to few centimeter and, in long term measurement, millimeters, but only for military purposes; when it comes to civil deployment GPS provides an accuracy of maximum 4.9 m [18]. The main limitations of GPS is that it is suitable only for outdoor use since in indoor environment signal is not able to reach the receivers.

The need of sensors able to track the position also indoor is the core characteristic of autonomous ground vehicles and they can be mainly divided in two different categories: Absolute and relative localization systems.

Relative position is a kind of localization dependent of previous state, which means

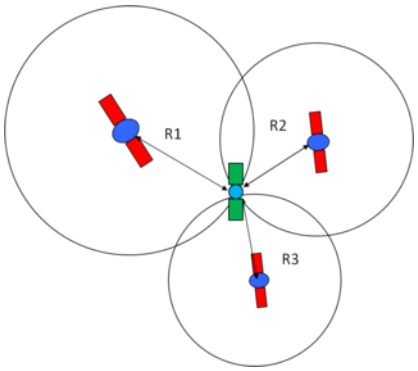


Figure 1.4: GPS Trilateration Mechanism

that the system does not know a priori its position but it knows only its local localization relative to previous poses, while absolute position, as can be intuited by term is referred to the global localization performed with external resources. Usually relative positioning is made possible by on-board sensors like encoders,gyroscopes, inertial measurement units etcetera, these techniques usually suffer of poor precision on the long time scale since usually they accumulates error with the passing of time. On the other hand global positioning rely on extern sources like beacon, GPS, or landmarks[19]. Descriptions of the systems to build maps of unknown environments usable to perform global positioning methods will follow as long with the methods to detect obstacles and key feature on the surrounding.

Camera

Among various sensors Cameras can be used not only to localize a AGV but also to extract key elements and useful information from the recorded images[20].
As far as localization is concerned Cameras, single or stereo, are at the base of the visual Simultaneous Localization and Mapping (vSLAM). Images can be used to save the map of an unknown environment in order to globally position a vehicle or they can be used to track the motion of the robot by subsequent reference frames transformations, each of them related to the previous state. In chapter 2 an exhaustive explanation of this technique will be carried on. The use of cameras is not limited just to localization but they can be exploited to have the perception of the depth

and extract from it important information regarding obstacles or goal objects. Last but not least from the images of the camera the robot can be made able to discern between the nature of different objects and take decisions accordingly[21].

LiDAR

LiDAR stands for Light Detection and Ranging and it is an efficient method to scan surrounding environment in two or even three dimensions[22]. These technology works in a fashion similar to the one of the Radar except from the fact that it exploits pulses of light in place of radio waves. Performing what is called *Time of Fights Measurements*, LiDAR calculates the time it takes for the light to reach the obstacle and hit back the emitter, in this way this technology senses the distances from objects occuping the space[23]. In case of three dimensional scanning, LiDAR can be used to build point cloud based maps from which different objects shapes can be recognized, moreover they reveal to be useful in those application requiring view under the surface, for example crack detection or surface degradation.

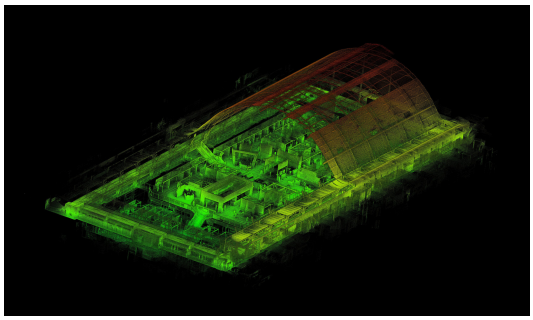


Figure 1.5: Example of an output map from LiDAR inspection

1.2.2 Interaction with environment

In service robotics the interaction is not limited with the environment, but often also human final users are involved, to this aim the machine must be able to effectively interact in order to complete its job. The way robots like AGV operating in stocks or

charging areas work is closely related to their end effector. The nature of this tool, usually located at the extremity of the manipulator arm, is directly related to the final scope of robot itself. Just to make some examples the types of end effectors varies from a gripper or robotic hand to a welder or pressure sensor.

Figure 1.6: Sample images of end effectors from Robotiq®[3]

This kind of approach is suitable when the only variable to take into account is the efficiency of operations, but when it comes to deal with human more attention must be paid in the interaction design. The concept of humanoid robot acceptance by human was firstly studied by Professor Masahiro Mori and carried to the discovery of what has been named *Uncanny Valley*. This theory asserts is possible to reach a certain level of affinity only abandoning a human-like design in order to overcome the risk of falling inside the *Uncanny Valley*, this deflection shown in figure 1.7. From these studies[4] it appears that humans tends to accept and like robots only up to a certain degree of human likeness and the reject is made worse by the movement as figure 1.8 shows.

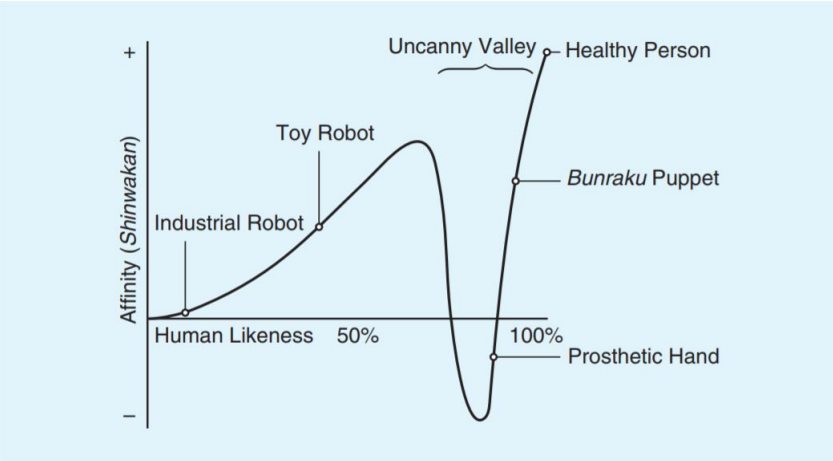


Figure 1.7: Uncanny valley[4]

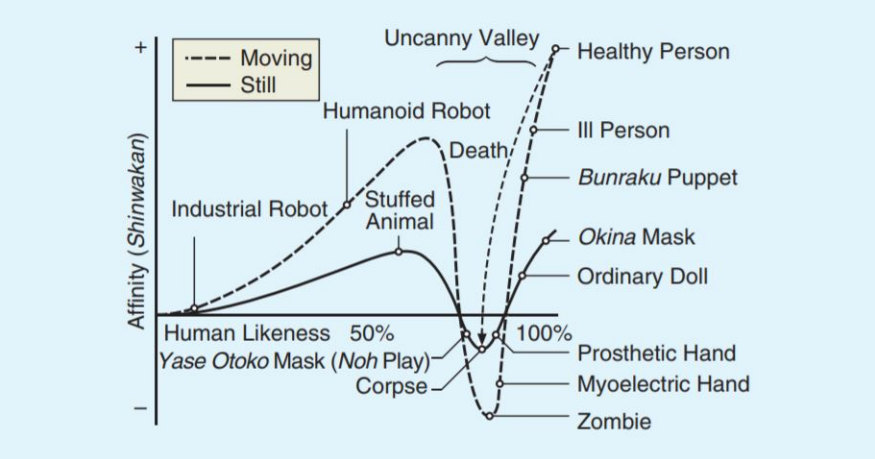


Figure 1.8: Effect of movement on Uncanny valley plot

An empirical prove of this trend can be found in the shape and appearances of commercial examples of autonomous robot that are designed to strongly and effectively interact with people such as LG Airstar® (1.1.2) or Savioke(1.1.1)®. Autonomous vehicles like those interacts with people through the use of tablets or voice recognition techniques and do not exploit end effectors since their services does not require a manipulation of the surrounding but only to carry loads,provide information and directions, and most important be able to localize and navigate in an unstructured,dynamical and indoor environment. The purpose of this work falls inside this last scope as will be exhaustively described in following chapter 3.

2 Simultaneous Localization and Mapping techniques

The aim of this paper is to describe the work done with the start up Innotech aimed to design and build a fully autonomous service ground robot. The design has been cured by the innovation incubator of San Diego airport, in whose environment the robot has been developed and tested. The elementary operation that a AGV must be able to perform is to go from point A to point B. This apparently trivial problem becomes more difficult the more it is deeply analyzed. The machine must be able to locate itself and know where point A or B are or recognize if it reached one of them. The path from initial to final point must be created and in real time modified if obstacles come across. The scope of this chapter is to provide a summary of state of the art of Simultaneous Localization and Mapping techniques to solve one of the previously described problems, precisely the localization. A revision of the literature brings to the conclusion that SLAM exploits two different kind of technologies used as sensors to perceive the world, LiDAR and Cameras. The above mentioned AGV embeds a stereocamera for the reasons that will be explained in chapter 3 and in order to describe how system has been implemented a revision of the state of the art of the visual SLAM approach will be made.

2.1 Visual SLAM

Simultaneous localization and mapping is the term referred to the problem of estimating a moving vehicle position in a unknown environment while building a map

of surroundings. This is of key importance for the achievement of fully autonomous robots.

At each time instant t the following quantities are defined:

- $\boldsymbol{x_t}$ is the vector containing all the poses information about the robot
- $\boldsymbol{u_t}$ is the control input applied to the robot at time instatnt $t-1$
- $\boldsymbol{m_j}$ is the vector containing the j^{th} landmark coordinates
- $\boldsymbol{z_{jt}}$ is the observation measurement of the j^{th} landamark at time instant t

The statement of the SLAM problem can be generally described by equation 2.1

$$P\left(x_t, m | Z_{0:t}, U_{0:t}, x_0\right) \tag{2.1}$$

2.1 describes the probability distribution of the actual pose of the robot conditioned by the initial conditions or position, coordinates of landmarks and history of control inputs.

2.2 Visual SLAM sensors

Visual based Simultaneous Localization and Mapping exploits cameras to perceive the environment. These kind of sensors seem to be very appealing for the final purpose since they are cheap,light,compact and low power consuming, moreover they are suitable to fairly detect stable features and extract useful information from the environment[24]. Before going into the details describing how Visual SLAM algorithms work an explanation of the principle underlying the system will be provided: depth reconstruction from images. All the different sensor in the market can be subdivided in three main categories.

Monocular cameras

Monocular cameras, or Monocameras in short, are sensors characterized by just one lens that capture frames providing no immediate and clear information about depth.

Figure 2.1: Example of MonoCamera: Firefly ® S [5]

The most immediate way to understand the distance of an object from the camera is the *triangle similarity method*:

> *Two triangles are said "similar" if their angles are all congruent and corresponding sides ratio are equal*

Bearing this in mind it is possible to calculate the distance of an object with known dimensions precisely from the frame:

$$D = W * F/P \tag{2.2}$$

Where D is the distance between sensor and analyzed object, W is the known width of the object, F is the focal length that is an intrinsic parameter of the sensor and P is the width measured in pixels.

Of course this formula does not serve SLAM purpose since segmentation of the image, object recognition and known dimensions are almost never possible conditions. Through the years diverse techniques have been used to extract the depth information from mono images, but they can coarsely classified in two main categories[25]:

- **SFM**: Structure-from-motion techniques exploit the physics of motion and perception to understand the distance of a moving object inside fixed consecutive frames

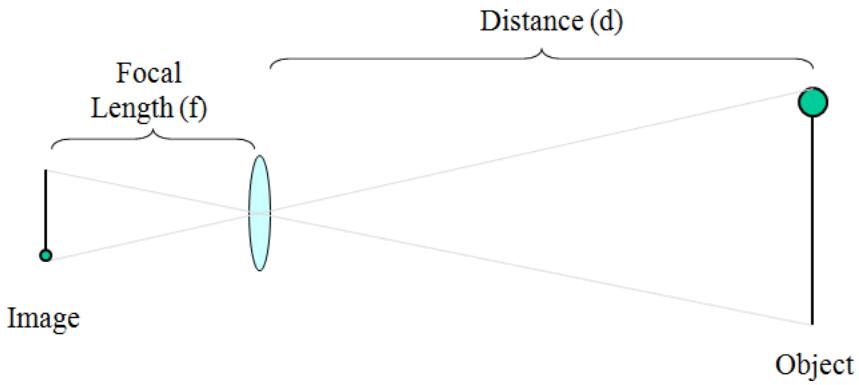


Figure 2.2: Distance calculation of known dimensions object

- **DFC**: Depth from Combining Defocus and Correspondence techniques relies on different depth cues such as texture,Focus/Defocus, occlusion or gravity

Tha main drawback of this approach aimed to convert bi-dimensional images to three dimension is that they are computationally expensive and not enough accurate for real-time, high frequency based applications[26]. When monocular cameras are used for SLAM purposes another approach is usually adopted to create a depth perception and it is the acquisition of the same image frame from different point of views to exploit the principle of triangulation, which will be described further on this chapter. Examples of pseudo-stereo implementations can be found in [27] [28] [29] [30] [31].

ToF Cameras

Figure 2.3: Example of ToF Camera: Basler ToF Camera[6]

ToF Cameras stands for Time of Flight Cameras, these sensors exploit the technology of Time-of-flight measurement, just like LiDAR, and provide several information

about the frame scene[32]. Basically these cameras rely on the principles of light reflection: the illumination unit emits modulated light with a solid state laser or LED operating near the infrared domain, about 850nm, and an imaging sensor, which accepts only same nature light, hit by photons converts energy in electric current. Phase shift between illuminating and reflected light is calculated an therefore the distance from the camera plane of each captured pixel[7].

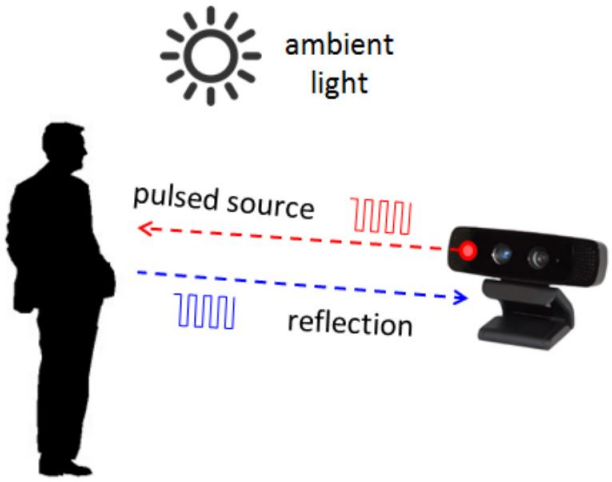


Figure 2.4: 3D time-of-flight camera operation (Image taken from Texas Instrument[7])

The advantage of ToF paradigm is its low software complexity, fast response time and independency from frame brightness, moreover range results to be scalable, from 10^{-2} to 10^{3} meters. Nowadays ToF cameras have low-resolutions and integrated solutions are not common[33], the solution is to use a regular RGB camera and a ToF sensor, therefore, calibrating the set-up, a final output 4-channel frame can be provided (RGB-Depth).

Stereocameras

The term Stereocamera refers to an hardware set up aimed to capture two images of the same scenario from two slightly different point of views in order to exploit the stereoscopic properties, which will be described further on. Basically a stereo system is made up by two identical lenses fixed at a certain distance called baseline.

Figure 2.5: Example of Stereocamera: ZED® camera[8]

Stereocameras usage relies on the principle of *Triangulation*, this is the process able to determine the three dimensional position of an object from multiple images, in this case two. The basic principle of triangulation can seem trivial when no source of errors and uncertainty are taken into consideration.

From figure 2.6 can be seen that the 3D point P represented in the real world refer-

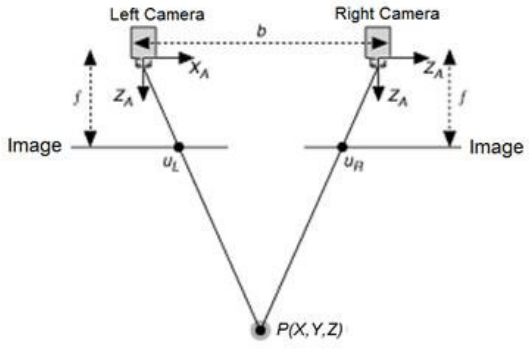


Figure 2.6: Simplified Stereo Vision System[9]

ence frame $\mathcal{W}(X, Y, Z)$ is projected on two different locations in the two dimensional camera planes and these can be computed by equations 2.3 and 2.4.

$$u_L = f \cdot X/Z \tag{2.3}$$

and

$$u_R = f \cdot (X - b)/Z \tag{2.4}$$

Disparity is now defined as the difference between 2.3 and 2.4

$$Disparity = u_L - u_R = f \cdot b/Z \tag{2.5}$$

And finally 2.5 is linearly related to the depth:

$$depth = f \cdot b/disparity \tag{2.6}$$

In reality a stereo set-up is more complex and even the best stereocamera will introduce some distortions in the output, although explained basic principles are still valid.

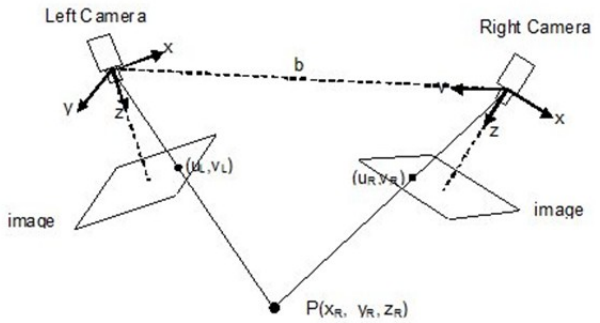


Figure 2.7: Typical Stereo Vision System[9]

The main arising problem to tackle are three:

- *Evaluation of intrinsic parameters:* The evaluation of the distortion parameters introduced by lenses is of key importance to understand the spatial relation between the two camera planes and hence a correct depth estimation

- *Rectification of captured images:* Images taken from the two cameras are not parallel and usually do not lie on a common plane, the final aim of rectification is exactly this, project images onto the common plane.

- *Point correspondence on both frames:* Last but not least, triangulation requires an exact correspondence of projected points, if this does not hold, even correct geometrical reasoning's would lead to incorrect results.

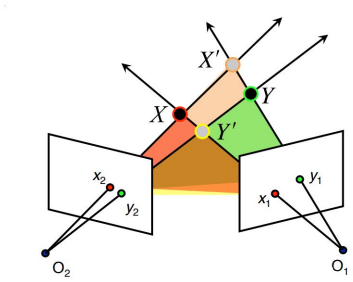


Figure 2.8: Incorrect triangulation example

From previous sensors description some reasoning will be made in order to explain why the choice of the sensor to embed in the project AGV has been taken. Monocular cameras were discarded due to their intrinsic criticalities in perceiving depth. Stereocamera has been preferred to ToF sensors because, as will be described in chapter 3, an artificial intelligence algorithm called YOLO has been deployed on the machine and it needed high resolution images to extract information from the frames. Moreover, in order to reduce the number of components of the system and in this way its complexity the choice has been to mount a ZED® stereocamera, which would serve both YOLO and SLAM purposes.
In the light of this, deeper description of stereo system based SLAM will follow.

2.2.1 Camera preparation for SLAM

As previously mentioned the first step to take when using a stereocamera is the calibration process, this is the starting point to obtain more precise and reliable depth measurements[34]. Scope of calibration is to estimate intrinsic and extrinsic parameters of the camera. Main source of distortions come from misalignment and assembling tolerances on mechanical sizes or lenses manufacturing defects.

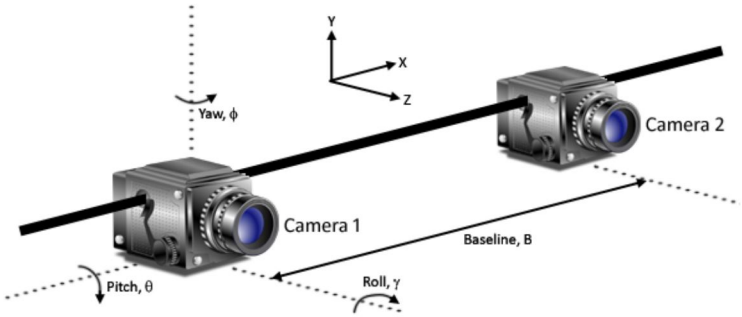


Figure 2.9: Stereocamera misalignment error sources[10]

In [35] the authors reports in order of relevance the most important parameters to be taken into account when setting up and calibrating a camera system. They are shown in table 2.1, sensitivity mathematical expressions are also reported.

An important source of error in stereo disparity calculation and rectification process arises from lenses geometrical and mounting imperfection. This last branch can be further subdivided in three main categories[36]:

- **Radial Distortion :** This causes set of points from the original image to move inward grouping together and scaling down dimensions or moving outward and spreading out towards margins. When distortion is negative it is referred as *barrel distortion*, when positive *pincushion distortion*. This type of distortion is symmetrical with respect to the optical axis and it is governed by the following equation 2.10, if perfect lens centering is assumed.

$$\delta_{\rho r} = k_1\rho^3 + k_2\rho^5 + k_3\rho^7 + ... \tag{2.7}$$

Table 2.1: Sensitivity to depth precision for different misalignment[10]

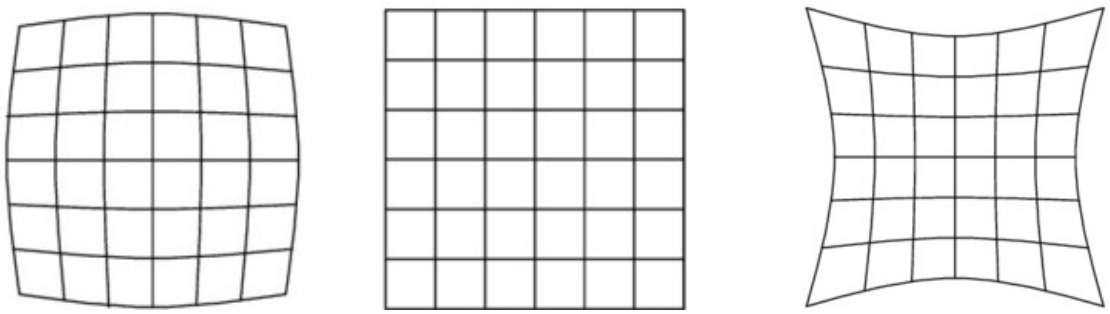
Error Source	**Relative Change in Depth**
Yaw Error $\Delta\phi$	$\frac{\Delta Z}{\Delta\phi} \approx -\frac{Z^2}{B}(1+X_2)$
Sensor Tilt$\Delta\phi$	$\frac{\Delta Z}{\Delta\phi} \approx -\frac{X_2^2}{B}$
Pitch Error$\Delta\theta$	$\frac{\Delta Z}{\Delta\theta} \approx \frac{Z^2}{B}X_2Y_2$
Roll Error$\Delta\gamma$	$\frac{\Delta Z}{\Delta\gamma} \approx \frac{Z^2}{B}Y_2$
Baseline ErrorΔB	$\frac{\Delta Z}{\Delta B} \approx -\frac{Z}{B}$
Focal Length ErrorΔf_2	$\frac{\Delta Z}{\Delta f_2} \approx -\frac{Z^2}{Bf_2^2}$

Where ρ is the distance from the principal point of the image plane and k_n are radial distortion coefficients. Equation 2.7 describes the point by polar coordinates (ρ, ϕ) and in particular the above mentioned imperfection will affect only radial displacements, in order to express the error in the same reference frame of the image a Cartesian coordinates transformation must be applied, as equations 2.8 and 2.9 show.

$$\begin{aligned} x &= \rho\ \cos\phi \\ y &= \rho\ \sin\phi \end{aligned} \tag{2.8}$$

$$\begin{aligned} \delta_{xr} &= k_1 x\left(x^2+y^2\right) + o\left[\left(x,y\right)^5\right] \\ \delta_{yr} &= k_1 y\left(x^2+y^2\right) + o\left[\left(x,y\right)^5\right] \end{aligned} \tag{2.9}$$

Usually radial distortion are mainly caused by flawed radial curvature of the lens elements.



(a) Barrel distortion (negative) (b) Undistorted object (c) Pincushion distortion (positive)

Figure 2.10: Radial distortion

- **Decentering Distortion :** This defect comes from the non collinearity of lenses optical centers and it introduces both radial and tangential point displacement. It is analytically described by equation 2.10.

$$\begin{aligned} \delta_{\rho d} &= 3(j_1\rho^2 + j_2\rho^4 + ...)\sin(\phi - \phi_0) \\ \delta_{td} &= (j_1\rho^2 + j_2\rho^4 + ...)\cos(\phi - \phi_0) \end{aligned} \tag{2.10}$$

Where ϕ_0 is the angle between the positive x-axis and the known maximum tangential distortion axis. Applying following transformation 2.11 and substitutions 2.12 2.13 to 2.11 it is possible to obtain 2.14.

$$\begin{pmatrix} \delta_{xd} \\ \delta_{yd} \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \delta_{\rho d} \\ \delta_{td} \end{pmatrix} \tag{2.11}$$

$$\begin{aligned} \cos\phi &= x/\rho \\ \sin\phi &= y/\rho \end{aligned} \tag{2.12}$$

$$\begin{aligned} p_1 &= -j_1 \sin\phi_0 \\ p_2 &= j1\cos\phi_0 \end{aligned} \tag{2.13}$$

$$\begin{aligned} \delta_{xd} &= p_1(3x^2 + y^2) + 2p_2xy + o\big[(x,y)^4\big] \\ \delta_{yd} &= 2p_1xy + p_2\left(x^2 + 3y^2\right) + o\big[(x,y)^4\big] \end{aligned} \tag{2.14}$$

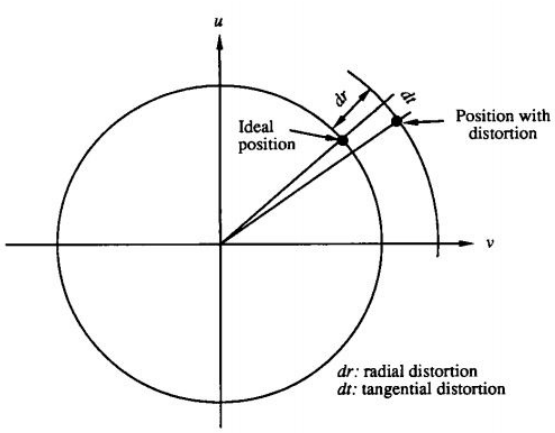


Figure 2.11: Decentering Distortion

- **Thin Prism Distortion :** This last sink of error arises from imperfect lens manufacturing and assembly leading to the adjunction of a thin prism to the optic projection and just like decentering distortion it causes both tangential and radial displacements. Thin prism can be mathematically modeled by 2.15 and with patterns similar to 2.11 2.12 2.13 follows 2.16.

$$\begin{aligned} \delta_{\rho p} &= (i_1\rho^2 + i_2 rho^4 + ...)\sin(\phi - \phi_1) \\ \delta_{tp} &= (i_1\rho^2 + i_2 rho^4 + ...)\cos(\phi - \phi_1) \end{aligned} \tag{2.15}$$

Where ϕ_1 similarly to ϕ_0 is the angle between positive x-axis direction and maximum tangential distortion axis, but as the two distortions have different effects on radial and tangential displacement the two maximum distortion reference axis can be different.

$$\begin{aligned} \delta_{xp} &= s_1\left(x^2 + y^2\right) + o\left[(x,y)^4\right] \\ \delta_{yp} &= s_2\left(x^2 + y^2\right) + o\left[(x,y)^4\right] \end{aligned} \tag{2.16}$$

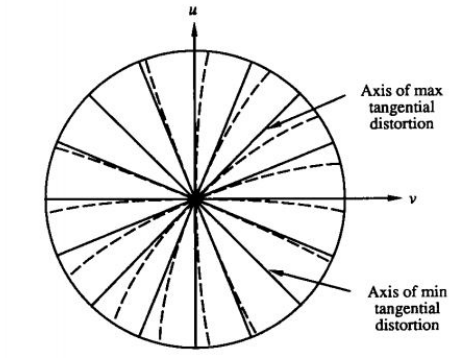


Figure 2.12: Effect of tangential distortion

Assuming negligible all the terms of order higher than 3 it is possible to superimpose the effect of the treated distortions in order to obtain the two comprehensive formulas for point displacement along X and Y directions. By summing equations 2.14 2.9 2.16 it is possible to finally obtain the equation 2.17.

$$\begin{aligned} \delta_x(x,y) &= s_1\left(x^2+y^2\right) + p_1(3x^2+y^2) + 2p_2xy + k_1x\left(x^2+y^2\right) \\ \delta_y(x,y) &= s_2\left(x^2+y^2\right) + 2p_1xy + p_2\left(x^2+3y^2\right) + k_1y\left(x^2+y^2\right) \end{aligned} \tag{2.17}$$

Different tools are available to calibrate the stereo pairs of cameras and obtain their parameters from simple procedures, examples can be OpenCV calibration camera library[37] and *Matlab Stereo Camera Calibrator App*[38]. Both of them will be further described in chapter 4.

2.2.2 Image processing for SLAM

Before explaining how to tackle images rectification and point correspondence problem, some definition are needed:

Epipole

> *The epipole is the point of intersection of the line joining the optical centres, that is the baseline, with the image plane. Thus the epipole is the image, in one camera, of the optical center of the other camera*

Epipolar Plane

The epipolar plane is the plane defined by a 3D point M and the optical centres C and C'

Epipolar line

The epipolar line is the straight line of intersection of the epipolar plane with the image plane. It is the image in one camera of a ray through the optical centre and image point in the other camera. All epipolar lines intersect at the epipole.

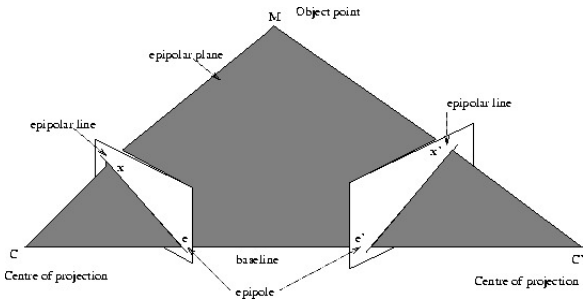


Figure 2.13: The epipolar constraint

Definitions and figure 2.13 have been taken from [39].
Assuming two cameras O and O' with fixed baseline b and linked by the transformation matrix 2.18:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.18}$$

point correspondence is the term referred to the problem of identifying which point x' on image plane O' corresponds to a point x on image plane O.
With reference to figure 2.14, the world point X is projected on image plane O in x that is the intersection point between ray OX and image plane O. In order to find x' it is possible to notice that ray OX has as image a line, called *epipolar line*, on image plane O' and that x' must lay on this line, in particular it is the intersection between ray $O'X$ and the image line of OX on image plane O'[40].

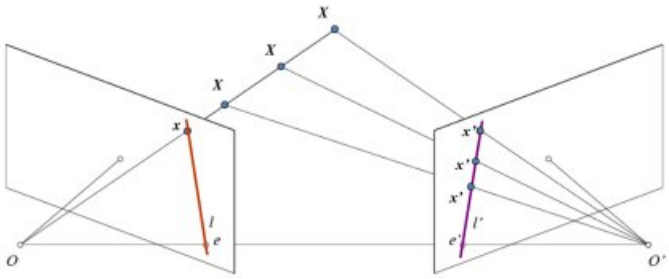


Figure 2.14: Epipolar geometry

These statements held because of some geometrical constraints, points O, O' *and* X form a plane called *Epipolar plane* and epipole e is image of camera center O' and vice versa. Same reasoning made for a point correspondence relation from O to O' remain valid also in the opposite case. It is worthy to notice that different world points generate different epipolar lines on image planes, but all of them intersect in the epipoles.

Now that the geometric problem has been stated it is possible to see how these points and line can be analytically found, in order to do this the concept of *Fundamental Matrix* must be introduced. The fundamental matrix **F** is used to translate the epipolar geometry into algebraic form[41].

$$l' = [e']_{\times} H_{\pi} x = Fx \tag{2.19}$$

Where l' is the epipolar line on image plane O', the notation $[e']_{\times}$ is defined as a 3x3 skew-symmetric matrix and H_{π} is a two-dimensional homography mapping each x_i to x'_i. Fundamental matrix results then to be defined in equation 2.20.

$$F = [e']_{\times} H_{\pi} \tag{2.20}$$

Must be pointed out that **F** represents a 3x3 mapping matrix of a two-dimensional to one-dimension transformations, hence it must have rank equal or lower to 2, generally it is 2. Moreover the geometric derivation described by equation 2.19 relies on a scene plane but it is not a necessary condition for the existence of **F**.

It follows that fundamental matrix possesses some properties describing what stated before.

Point Correspondence: Given two correspondent points

$$x'^T F x = 0 \tag{2.21}$$

Epipolar lines: Given two epipolar lines of two correspondent points

$$\begin{aligned} l' &= Fx \\ l &= F^T x' \end{aligned} \tag{2.22}$$

Epipoles:

$$\begin{aligned} Fe &= 0 \\ F^T e' &= 0 \end{aligned} \tag{2.23}$$

In order to estimate the fundamental matrix it is possible to take two projected points $x = (x, y, 1)^T$ and $x' = (x', y', 1)^T$ and solve equation 2.21 as follows in equation 2.24.

$$\begin{aligned} &x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \\ &Af = \begin{bmatrix} x'_1x_1 & x1_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 \end{aligned} \tag{2.24}$$

To the aim of finding a unique solution of **F**, matrix **A** must have rank 8 and in this case solution is trivial (up to scale), but due to noise rank of matrix **A** can become higher than 8, usually 9 since it has 9 columns. In the last case the Least Square estimator can be used in order to minimize $\| Af \|$ subject to $\| f \| = 1$.This approach is called *8-point algorithm*.

In case matrix A has rank 7 thus having dimensions 7x9, still the problem is feasible exploiting the singularity constraint of **F** leading to a unique solution (up to scale), this last method is referred as *7-point correspondences*[42].

In order to reduce the degrees of freedom of the estimation problem it is possible to introduce the specialization of **F** to the case of normalized coordinates called *Essential matrix* **E**. Knowing the calibration matrix **K** it is possible to decompose the camera matrix as follow: $P = K\,[R|t]$ resulting in the relation 2.25.

$$x = PX \tag{2.25}$$

Normalized coordinates are then defined as the result of the transformation 2.26

$$\hat{x} = K^{-1}x = [R \mid t]\,X \tag{2.26}$$

Considering now a stereo-pair with normalized camera matrices $P = [I \mid 0]$ and $P' = [R \mid t]$ the fundamental matrix takes the name of essential matrix and has the form described by equation 2.27.

$$E = [t]_{\times} R = R\left[R^T t\right]_{\times} \tag{2.27}$$

The defining equation for the essential matrix is $\hat{x}'^T E\hat{x} = 0$ and substituting 2.26 it follows that **E** and **F** are related by equation 2.27.

$$E = K'^T F K \tag{2.28}$$

The reduced degrees of freedom of the essential matrix with respect to the fundamental one result in extra constraints exploitable to solve the correspondence problem[42]. Having discussed the *correspondence problem* it becomes clear how it is the underlying layer of every possible SLAM algorithm and technique and because of this it must be tackled with care. In modern application it is a de facto requirement for almost every SLAM software to work with rectified images[43]. Rectification process applies to the images a transformation in order to project the two different image planes to a common plane so that epipolar lines of both frames are coincident and parallel to the coincident horizontal axis of right and left images. This process is essential for

stereo analysis since correspondence seeking algorithms can reduce the research space to only one dimension, speeding up execution time and accuracy of the final result.

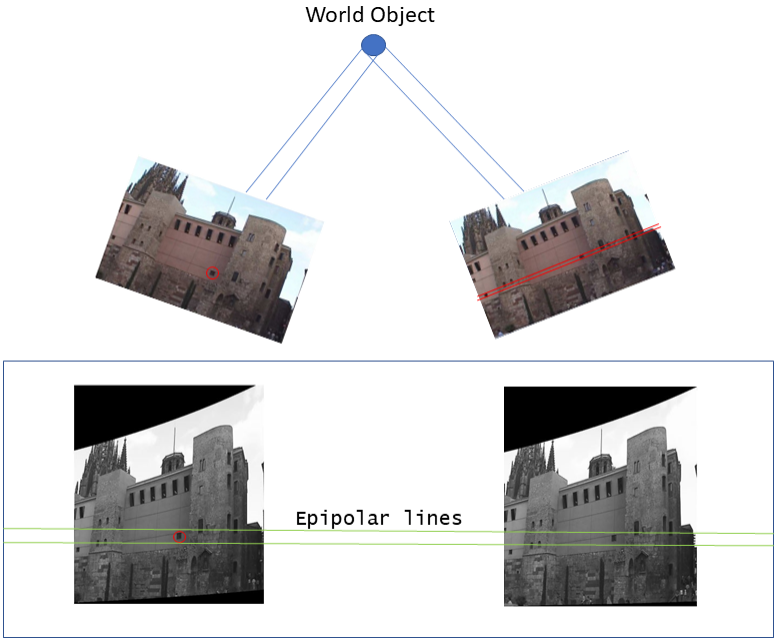


Figure 2.15: Rectification of an image stereopair

What stated before is the core of the matter and what traditionally has been done in order to rectify image stereopairs[44], but some problems arises when epipoles are inside or near to the frames space, in fact this makes the rectified images to go unbound or at least increase their dimensions. Even if a lot of different algorithms and methods are now available in literature[45][46], these results to be complex and not always the most effective. Chen et al.[43] proposed an easier method that relies only on the fundamental matrix. The input of the algorithm is a stereopair with an overlapping region, from this it is possible to find at least seven correspondent points for the reasons discussed in 2.2.2, find **F** and the correspondent epipoles in the two images and exploit the epipolar constraint to find the overlapping region of the stereopair. Assuming now the left image to have pixels dimensions nxm and corners coordinates $A(0,0,1)^T$, $B(0,n,1)^T$, $C(m,n,1)^T$, $D(m,0,1)^T$, epipolar lines are calculated with the formulas 2.29.

$$\begin{aligned} l'_1 &\sim \mathbf{F}A \\ l'_2 &\sim \mathbf{F}C \\ l'_3 &\sim \mathbf{F}^T A \\ l'_4 &\sim \mathbf{F}^T C \end{aligned} \tag{2.29}$$

The equivalence is not strict since, as discussed previously, correspondence problem is determined only up to a scale factor and 2.29 hold only for a non null scale factor. Having found the epipolar lines it is finally possible to extract the pixels value with the *Brasenham Algorithm* and resample the rectified images.

2.3 Visual SLAM methods and implementations

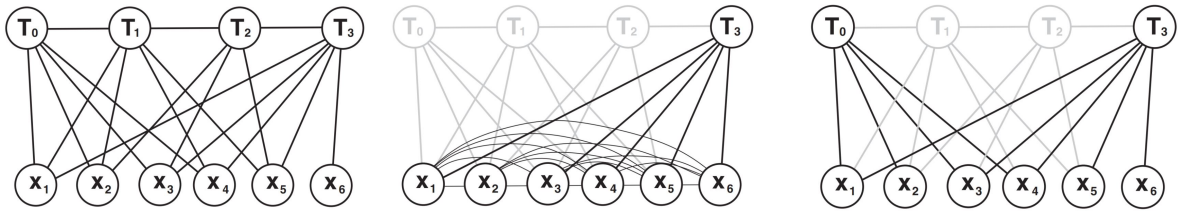
In section 2.2 different technologies of visual perception as well as the theory background underneath the main approaches of depth perception from two-dimensional images have been described, from these it is possible to start introducing how Simultaneous Localization and Mapping can be implemented.

The literature describing SLAM implementations is various and comprehends diverse approaches and methodologies, in fact Simultaneous localization and mapping has not reached an end point but instead is still object of deep researches and improvements. This section of the work aims at describing the principle fashions of methodologies at the base of SLAM algorithms. According to Strasdat et al.[47] SLAM and SFM, whose principles were derived by photogrammetry, are similar problems aimed at the same scope: the estimation of a sensor pose moving inside an unknown environment, just motivated by different means. The latter wants to reconstruct off-line a 3D space from a relatively small set of image batch, while the former aims at tracking the motion of a sensor, camera or LiDAR, in real-time. From this distinction of the two it becomes clear that in SFM the final result quality overcomes the time to achieve it, while in SLAM this is not always true because of the requested firm frequency of the running system.

Focusing on SLAM approaches two methodologies have prevailed on others: filter-

ing approaches, which update probabilistic distributions by merging information in a sequential fashion from all the previous images; and bundle adjustment (BA) that are oriented towards heuristically select specific images, called keyframes, evenly distributed in space and optimize the obtained batch reaching absence of drift errors in long runs.

With reference to figure 2.16a it is possible to think about SLAM as a graph and analyze the inference with a Markov random field, where T_i represents the positions covered by the camera as a vector and static features are represented by vector x_j, while the lines in the graphs stand for the observations of the j^{th} feature from the i^{th} pose.



(a) Markov Random Field (b) Filter (c) Keyframe BA

Figure 2.16: (a)SLAM/SFM as Markov random field. (b) and (c) Inference propagation in filter and keyframe approach

It is clearly visible that as the real time operation goes on the dimension of the sets of positions and features increase inevitably since new keyframes would be inserted as new paths are covered. The solution to the SLAM problem involves the resolution of the Markov graph from scratches while it is growing step after step, but this is not feasible because of the computational cost growth at each new frame, considering that it increases linearly in keyframe BA based approach as $O(N)$ and cubicly in filtering as $O(N^3)$. Considering the estimation of the state T_i provided by the filtering solution it can be noticed that each feature recognition is subordinated to previous states of the system. This method gains its consistence by holding the information of the features that are likely to be measured again in the future, but this results in a strong correlation between states. The final outcome graph results to be compact

and not likely to spread exponentially as new features are added when new areas are investigated. The drawback is that removing or updating a previous state would cause changes in all subsequent ones causing the increase of computations and justifying the cubic increase with measurements. The standard filter used is the *Extended Kalman Filter (EKF)* even if also mixed approaches exploiting particle filtering are common[24]. On the other side there is BA optimization which solves the graph from scratches at each new frame getting rid of the grater part of the past poses except for the keyframes. Graph in 2.16c results to be more filled with previous position vectors but lowly inter connected among them, resulting in a solution efficiently optimizable. The two described philosophies are very different, one is getting advantage of compact but complex routines to manage graph (filtering), while the other (BA) is exploiting a verbose but highly efficient database of past poses.

As Strasdat et al.[47] pointed out, more mature implementations involve keyframe BA based approaches in perfomring SLAM due to their better performances and scalability.

SLAM implementation can be divided in two steps, the feature extraction and the technique itself. The starting point is to use the images taken from camera to extract features from the frames. When the term feature is used in the SLAM field it is usually referred to *points features*, but as Naveed et al. pointed out[24] there are methods that exploits *line/edge features* because of the main advantage of being invariant to light conditions and moreover, they provide useful information about geometry of surroundings. There are also examples of featureless SLAM implementations[48] where motion has been tracked with no feature extraction from the image but just by considerations on pixel values inside the central sub-window of the frame.

As previously said the most mature approach in SLAM implementation is the point feature extraction and this procedure involves a image processing step that can be carried out using different algorithms which peculiarly characterize different SLAM developments[49].

2.3.1 Feature extraction algorithms

A feature is defined as a specific information extractable from an image and they are categorized in two ways:

- *Keypoint*, also called corners, are static features localized in a spot on the frame and characterized by patches of pixels around the coordinates. They present gradient change along all directions.

- *Edges* are particular feature characterized by their orientation or profiles, and because of this, they do not present gradient change along orientation. They are usually exploited as markers of boundaries between objects

Even if the different algorithms that will be further described in this section use completely different approaches the basic functioning scheme of feature extraction is always the same: Detection of interest point, description of feature and its matching among two different time instant images.
The identification of an interest point is made by examining itself properties; interest points are particular pixels spaces were boundaries of objects changes suddenly or were two different edge segments meet, and because of this they are stable under different light exposure on a local and global scale, or more in general, stable in case of perturbations of the landscape. In order to be classified as interest point it should be detectable repeatably and efficiently. Having identified a certain number of interests points or feature in an image, it is necessary to distinguish one from the others, and in order to this, it is possible to generate descriptors for each of them including information about shape or appearance aimed at recognizing the same point in two different images of the same object for example taken from two different perspectives or under two different light conditions. Descriptors can be both focused on local or global image domain. Having found the interests point and labeled them with descriptors, working as fingerprints, the last step is match interest point pixels coordinates between two different images.

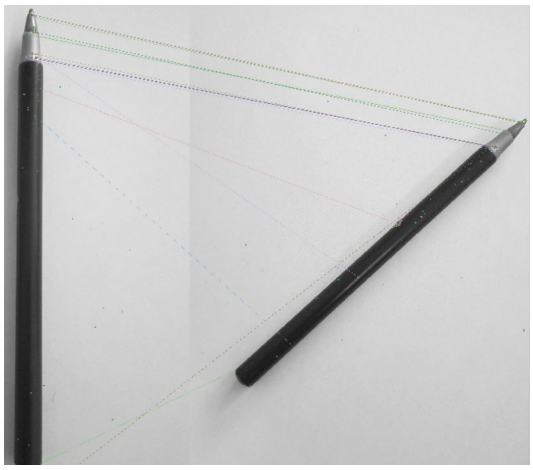


Figure 2.17: Feature matching

Features from Accelerated Segment Test[50]

Feature from Accelerated Segment Test, in short FAST, is an algorithm developed by E. Rosten and T. Drummond in 2006, it was born to solve the high computation cost of other feature detectors and, indeed, it is characterized by high computational efficiency even improved if machine learning is used as aid to the visual engine. For these reasons it is recommended for real-time implementations. The algorithm flow is now explained:

- Considering the pixel **p** let $\mathbf{I_p}$ be its intensity, the aim is to decide whether it is or not an interest point
- Choose a suitable threshold **t** and isolate a Bresenham circle of radius 3, or in other words individuate a 16 pixels rounded contour around **p**
- **p** is considered an interest point if at least **n** adjacent pixels are outside the intensity range $\boldsymbol{I_p - t < I < I_p + t}$
- Repeat for each entry of the image matrix

To speed up the algorithm is possible to exploit some tricks in order to avoid useless checks of all the 16 pixels. It is possible to reject as interest point the pixel **p** if less than 3 of pixels 1,5,9 and 13 fall into the above mentioned range. If this first test is passed it is possible to check for all the other 12 pixels around **p**, in the case **n**

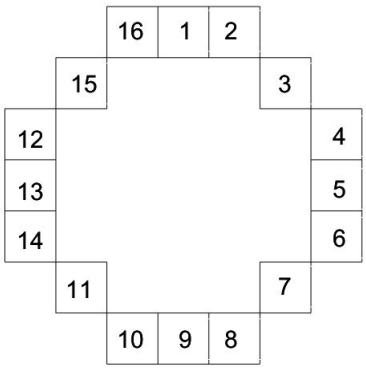


Figure 2.18: Bresenham circle

are contiguously found the point can be marked as of interest. The limitations of this algorithm concern the dependence of the execution time from the order pixels are queried, moreover setting of parameters such as **t** or **n** is not an easy task if they remain static among different sessions, because of all these considerations machine learning approach is adopted to optimize the software execution and performances. Machine learning improvement works in the following way:

- A feature vector $\mathbf{P}$ is determined including all 16 circle pixels around a feature point and divided in 3 subsets: $\mathbf{P_d}$ for darker points, $\mathbf{P_s}$ for similar points and $\mathbf{P_b}$ for brighter ines
- Defined a variable $\mathbf{K_p}$, which is true if the point is of interest or the opposite the other way, use the ID3 algorithm on each subset. ID3 algorithm works minimizing the entropy of the system.

$$\begin{aligned} H(P) &= (c + \bar{c}log_2(c + \bar{c}) - clog_2c - \bar{c}log_2c \\ where\ c\ &=\ |\{p|K_p\ is\ true\}|\ (number\ of\ corners) \\ where\ c\ &=\ |\{p|K_p\ is\ false\}|\ (number\ of\ non\ corners) \end{aligned} \tag{2.30}$$

Another limitation of the above described algorithm is the problem of keypoints concentrated density, in fact, if no actions are taken a lot of keypoints will be found in the same local area, figure 2.19.

Figure 2.19: Image without points suppression

A possible solution could be to evaluate the difference in intensity of pixel **p** with respect to the ones belonging to the Bresenham circle and score the interest point accordingly, only the points with highest score will be used, figure 2.20

Figure 2.20: Image with points suppression

It is interesting to point out that FAST algorithm surprisingly does not work well with perfect images produced on purpose by computer since crisp images tend to fail the corner test. Computer-generated images must previously be processed and blurred, for example with a Gaussian filter, in order to transform sharp corners in less precise edges.

Harris corner Detector[51]

Harris corner detector has been introduced by C. Harris and M. Stephens in 1988 and it is characterized by the differential calculation along directions, since 1988 it has been strongly improved and used in many machine vision implementations because of its ability to distinguish corners and edges.
The idea at the base is to investigate those sub windows whose pixels are unique and this can be made comparing the points values of a window with adjacent ones in all the 8 directions. *Sum Squared Differences (SSD)* is taken as criteria and evaluated for each shift, so let $E(u, v)$ be the sum of all SSD of a 3x3 window.

$$E(u,v) = \sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 \tag{2.31}$$

Corner detection problem is a maximization problem of functional $E(u, v)$ and in order to do that it is advisable to maximize the difference, it follows:

$$\begin{aligned} & E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \\ & with: \\ & M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & Y_y^2 \end{bmatrix} \end{aligned} \tag{2.32}$$

Solving for the eigenvectors of **M** it is possible to score the gradient and discern between flat regions, corner or edges (2.2).

$$R = detM - k(traceM)^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{2.33}$$

Table 2.2: Harris detector eigenvalues criteria

$\|R\| \sim 0$	$\lambda_1 \sim 0$ and $\lambda_2 \sim 0$	FLAT REGION
$\|R\| < 0$	$\lambda_1 >> \lambda_2$ *or* $v.v.$	EDGE
$\|R\| >> 0$	$\lambda_1 \sim \lambda_2$ and $\lambda_1 >> 0$	CORNER

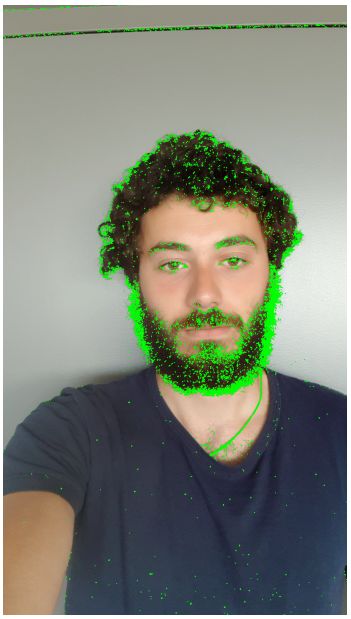


Figure 2.21: Harris detector output image

Scale Invariant Feature Transform[52]

Scale invariant Feature Transform (SIFT) has been developed in 2004 by D. Lowe, and it is a proprietary software of University of British Columbia. With respect to the previous one described algorithms this one embeds not only keypoints detection but also description and matching. This algorithm take into account the scale of the image, the represented object could seem big from a near point of observation but

also small if seen from far away, that is why images scales are separated into octaves, each one being the half of the previous, by simply reducing the pixels dimensions of the photo keeping the sides ratio constant. In this way information, scaling down, are definitely lost and operation cannot be reverted just by scaling up since resolution would not be the same.

Figure 2.22: Example of successive pyramidal images

Each octave is then progressively blurred with the Gaussian Blur operator. This is a convolution process of the image and the Gaussian operator. Each pixel is so treated with relation 2.34.

$$L(x, y, \sigma) = G(x, y, \sigma)I(\dot{x}, y) \tag{2.34}$$

Where I(x,y) is the pixel location in the image reference frame and G, the Gaussian operator, is defined as follows 2.35.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{2.35}$$

Figure 2.23: Example of successive application of Gaussian blur operator

Before analyzing the image in order to find the keypoints a new set of images is generated by subtracting two differently blurred images of the same octave and repeating the pattern for each octave. The term referred to the new set is *Difference of Gaussian kernel* (DOG). DOG can be seen as approximation of the Laplacian operator and the former is preferred since it is scale invariant, less computationally expensive and moreover Laplacian is really sensitive to noise.

Figure 2.24: DOG on the left and Laplacian operator on the right

After the above described step the effective keypoints are investigated comparing each pixel with its 3x3 neighborhood in the same, magnified and reduced subsequent scale

for a total amount of 26 comparisons. If the analyzed pixel is a local maximum or minimum is it labeled as possible interest point. When operation is concluded a lot of matches come out, so they are filtered by saving only those above a certain intensity threshold and rejecting the edges with an Hessian matrix acted to discover principal curvature. Having computed location of interest points it is possible to proceed with the orientation which is individuated by studying the gradient magnitude and orientation. Gradients are calculated in the surrounding of the keypoints and subdivided by magnitude. Higher magnitude gradient will tell the orientation of the keypoint. In order to build a descriptor it is taken a 16x16 mask around the pixels and the gradient evaluation operation is repeated for each 4x4 sub-block. Big intensity values are eliminated to guarantee illumination independence, while keypoint orientation is subtracted from the direction of the descriptor gradient in order to reach rotation independence. Keypoints are finally matched analyzing their nearest neighbors.

Speeded-Up Robust Features[53]

Speeded-Up Robust Features (SURF) was firstly introduced by H. Bay in 2009 and it relies on fast-Hessian detector for feature detection, it is patented as because of this not freely accessible. What differentiate it from a standard Hessian procedure is that this algorithm exploits integral images to boost computational efficiency. Integral calculations on images are efficient ways to sum or average intensities of pixels in a rectangular area with extremities $O(0,0)$ and $\boldsymbol{X}(x,y)$.

$$I_{\Sigma}(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j) \tag{2.36}$$

The detector is based on Hessian matrix calculation whose entries are the convolution of Gaussian blurring operator and pixel value.

$$\mathcal{H}(\boldsymbol{X},\sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{X},\sigma) & L_{xy}(\boldsymbol{X},\sigma) \\ L_{yx}(\boldsymbol{X},\sigma) & L_{yy}(\boldsymbol{X},\sigma) \end{bmatrix} \tag{2.37}$$

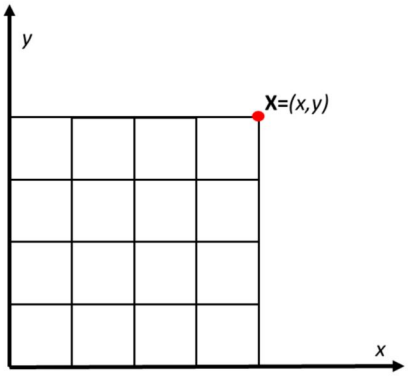


Figure 2.25: Integral images area of interest

Differently from SIFT where Laplacian of Gaussian(LoG) were approximated by DOG, SURF exploits box filters in order to simulate the Gaussian blurring, obtaining an approximation of the Hessian matrix determinant fast to compute and suitable for real-time applications and objects recognition. Images are also in this case divided in octaves and repeatedly blurred. In order to build the descriptor the orientation of the keypoint must be investigated. To do so, Haar-wavelet responses are calculated in x-y directions in a 6σ radius circle with a sampling time equal to σ. The orientation is understood individuating the direction where the responses sum is higher.

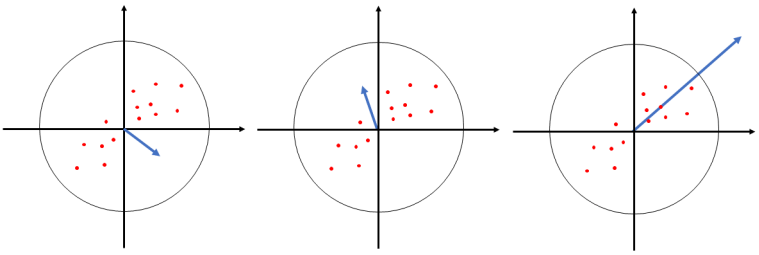


Figure 2.26: Haar-wavelet responses based orientation extraction

Similarly to the SIFT the approach is then repeated in each 4x4 sub-window of a square contour appositely built with dimensions 20σ. The so obtained descriptor vector is unique for each sub-window and has dimension 64, while in SIFT it had 128 entries.

Binary Robust Independent Elementary Features

Binary Robust Independent Elementary Features(BRIEF) is not a feature detector but just a way to build a descriptor vector. Algorithms like SURF or SIFT are very robust but their descriptors result to be quite time and resource consuming to be determined, that is why BRIEF has been designed. BRIEF works at a singular pixel level in order to build a binary word of variable length, from 128 to 512 bits.

$$V_n = [01000110101110...b_n] \tag{2.38}$$

The descriptor word is created by comparing pairs of pixels with a function defined by 2.39.

$$\tau(I, x, y) = \begin{cases} 1 & \text{for} : p(x) < p(y) \\ 0 & \text{for} : p(x) \geq p(y) \end{cases} \tag{2.39}$$

Considering a patch of sides nxn, the pairs to be compared can be choosen accordingly to different geometries (figure 2.27):

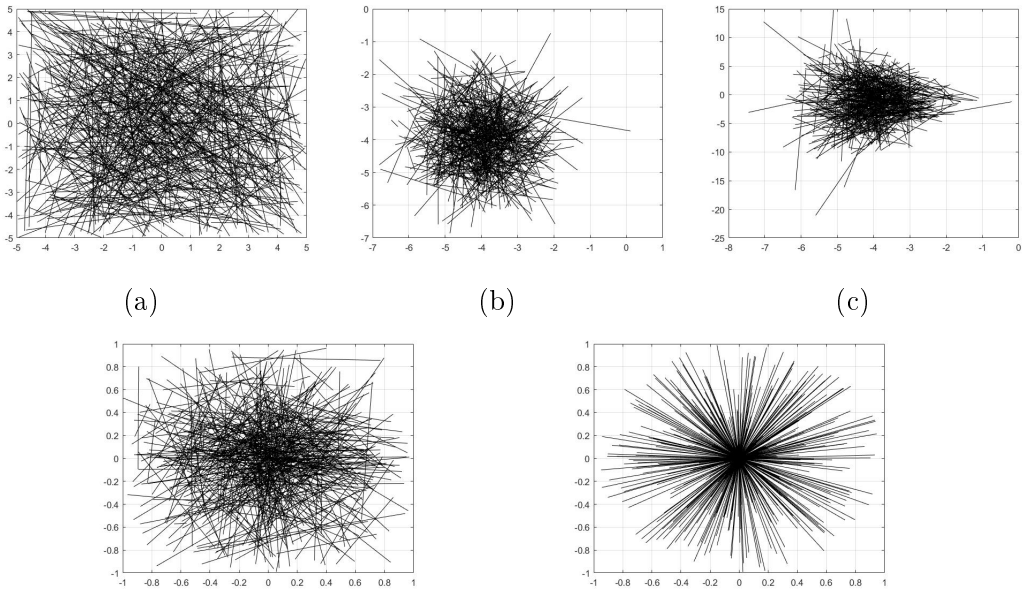
(a) Pairs can be extracted from a uniform distribution spreading around the keypoint in a range of $n/2$

(b) Pairs can be extracted from a gaussian distribution with $\sigma = 0.04 * S^2$

(c) A variant of point 2 acted to draw in a more keypoint centered neighborhood is to extract the two pixels from two different gaussian distributions characterized by: $\sigma_1 = 0.04 * S^2$ and $\sigma_2 = 0.01 * S^2$. First distribution is centered in the keypoint, second one in the first extracted pixel

(d) Pairs can be extracted from a series of concentric circles radiating out from the pole

(e) One point of the pair can be extracted from a series of concentric circles radiating out from the pole, while the other remain fixed to the pole



(a) (b) (c)

(d) (e)

Figure 2.27

In conclusion the BRIEF descriptor function will look like 2.40

$$f(n) = \sum_{1<i<n} 2^{i-1}\tau(I, x_i, y_i) \tag{2.40}$$

Oriented FAST and Rotated BRIEF[54]

Oriented FAST and rotated BRIEF (ORB) is the algorithm developed by OpenCV in 2011 to substitute SIFT and SURF, which are patented and not free to use. It uses FAST algorithm (described in 2.3.1) for keypoints detection, but improving it, exploiting pyramids level images to assign to each interest point an orientation, making actually FAST scale invariant.

To the aim of finding the orientation of a patch around a keypoint it is possible to perform following calculations.

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{2.41}$$

Having found the moments, equation 2.41, it is possible to calculate their center of mass.

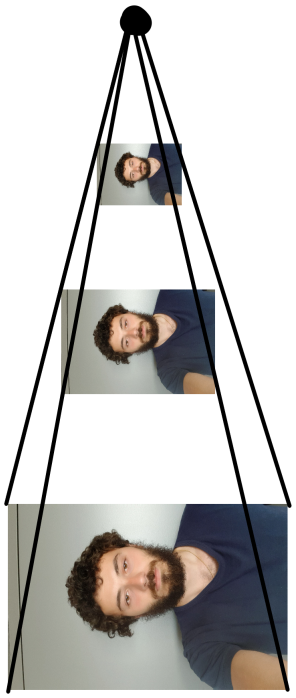


Figure 2.28: Pyramid levels of same image

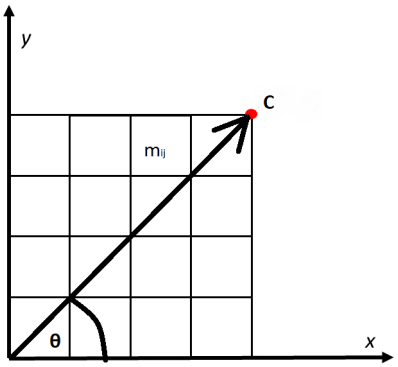


Figure 2.29: Orientation of interest point patch

$$C = \left(\frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}, \frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i} \right) \tag{2.42}$$

Orientation of the patch is now indicated by the vector $\overrightarrow{OC}$

$$\theta = atan \left(\frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i x_i} \right) \tag{2.43}$$

Once feature points are detected they are described by a slightly modified version of the BRIEF descriptor (equation 2.40), in fact the above mentioned descriptor presents really bad performance in rotation bigger than few degrees. In order to overcome the problem $2Xn$ matrix is built and rotated of the amount described by equation 2.43.

$$S = \begin{bmatrix} x_1, & \dots & x_n \\ y_1, & \dots & y_n \end{bmatrix} \tag{2.44}$$

$$S_\theta = R_\theta S$$

Figure 2.30: ORB extraction and matching

2.3.2 Keyframes selection and tracking

After having described how features can be extracted and matched from image to image it is possible to see how SLAM techniques work in order to track the camera pose and concurrently build a reliable map of the environment. Taking as example a standard hardware set up, it is not strange to think about a stereo sensor capturing 15 to 60 frames per second. They are too much information to be processed by an embedded computational unit. SLAM software have the capability to select some relevant frames to be processed, analyzed, described and used by the map generation thread. As multiple times it is read in the literature, SLAM is a chicken-or-egg problem because for mapping, a pose estimate is needed and for tracking the camera, the map is needed.

The first thing to be tackled is the reconstruction of the camera position by analyzing frame by frame. Features are extracted and described by the algorithms described in 2.3.1, then their 3D coordinates $\boldsymbol{X_j} = [x_j y_j z_j 1]^T$ are obtained from the 2D projections

$x_{ij} = [u_{ij} v_{ij} 1]^T$ exploiting triangulation and point correspondence explained in 2.2.2.

$$\begin{aligned} x_{ij} &= P_i \boldsymbol{X_j} \\ where & \\ P_i &= K\,[R_i\ t_i] \end{aligned} \tag{2.45}$$

In equation 2.45 matrix K contains all the calibration parameters of the camera describe in 2.2.1 and $[R_i\ t_i]$ is the roto-translation matrix describing camera pose in the world reference frame $\mathcal{W}$. It must be pointed that equality in 2.45 is ensured by the three-dimensional information introduced by the stereo baseline length, if this would not be true, location of landmarks would be determined up to scale and equation 2.45 would be:

$$x_{ij} \sim P_i \boldsymbol{X_j} \tag{2.46}$$

The estimation of a calibrated camera pose from at least 3 3D points coordinate can be referred as *Perspective-n-point problem* (PnP). PnP problem take the name P3P problem when pose estimation starts from the observation of three different world points.

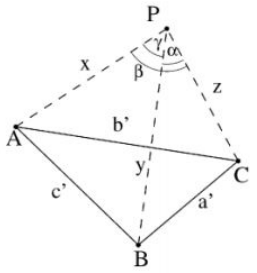


Figure 2.31: Perspective-3-points problem

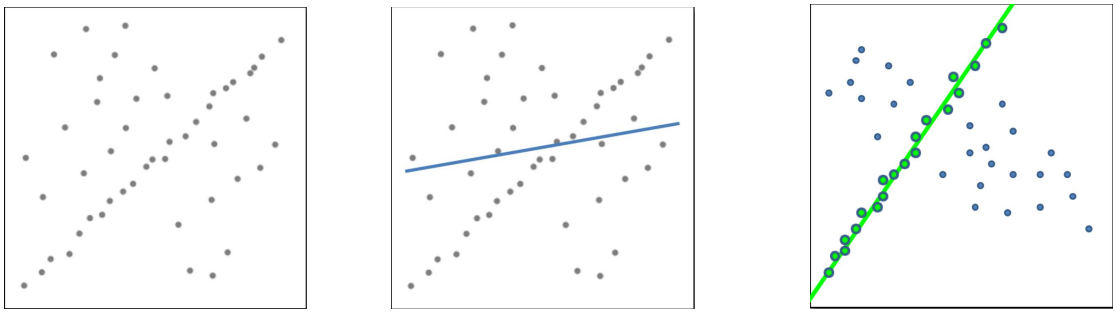
With reference to the figure 2.31 let $\overline{PA} = X$, $\overline{PB} = Y$, $\overline{PC} = Z$, $a = \overline{BC}$, $b = \overline{AC}$, $c = \overline{AB}$ and α, β and γ respectively the angles between $\overline{PB}$ and $\overline{PC}$, $\overline{PC}$ and $\overline{PA}$,

$\overline{Pa}$ and $\overline{PB}$, it possible to write the equation system for the problem.

$$\begin{cases} Y^2 + Z^2 - 2YZ\cos\alpha - a^2 = 0 \\ Z^2 + X^2 - 2XZ\cos\beta - b^2 = 0 \\ X^2 + Y^2 - 2XY\cos\gamma - c^2 = 0 \end{cases} \tag{2.47}$$

The system yields either infinite or at most 4 possible solutions. Due to this ambiguity in the results, PnP problem are usually solved with more than 3 world points, even if this carries increased complexity in solution, as well as increased computational cost[55]. Another approach not exploiting the PnP algorithms is based on the minimization of the reprojection error of 3D objects location and camera parameters[56]. This is usually done by building a cost function that change from implementation to implementation and find the minimum solution for the reprojection error, which is the error committed when calculating a three dimensional coordinate from its two dimensional projection, example of this application can be found in [57] [58] [56]. It is worth to notice that the minimization problem cannot be solved for the whole batch of feature points matched between two different temporal frame because of the presence of false positive matches generated by the native local tendency of descriptors. Before feeding the pose estimator with coordinates and points true features must be determined. Due to the presence of outliers, linear estimator like Least Squares proved to be inefficient. Just to make an example RANSAC optimization algorithm is often used in this field, even though its iterative nature it resulted to be suitable for real time applications [56][59].

As mentioned above, SLAM problem involves at the same time localization and mapping, which is why two different threads are running on the system, one for tracking and the other for map generation and optimization. Keyframes are those frames accepted to build the depth map and their selection varies from implementation to implementation.



(a) Batch of points with presence of outliers

(b) LS estimator

(c) RANSAC estimator

Figure 2.32

The most common methods are the following:

- Steepest descent like algorithms that insert keyframes in an initially empty batch only when they reduce the most energy. This is a cost expensive method with respect to others, in fact number of operations scale as $O(n^2)$[59]
- Close/far points threshold can be used to estimate the need to insert a new keyframe, in fact a certain number of close points (distant less than n times the baseline, where n can very from application to application) are needed to estimate translation. When the amount of detected close points is too low it is possible to add a keyframe[60]
- A shift threshold can be used as discriminant in keyframe insertion, in fact if a certain amount of roto-translation has been detetcted a new keyframe can be added[61].
- A last rather straightforward way to do so is to investigate the number of detected and matched features point, if they go down a certain threshold the frame can be labeled as keyframe[58].

Even if a keyframe database, working as a rudimentary map, has been created, comparing each new frame to each component of the database for localization purposes would be so onerous to make it impossible. In order to tackle this problem, keyframes are usually synthesized in a binary vocabulary, easy and fast to be read, organized

like trees. Keyframes, specially if recorded for long operations, can be very different one from the other and most of the times deep comparisons are not necessary, for example when the scenes are totally different. To speed up the process and make it more efficient a tree-like structure made of nodes is built and each node is scored with its level of distinctiveness.

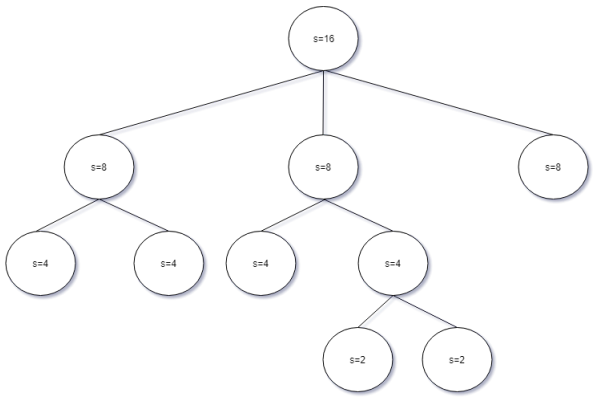


Figure 2.33: Tree-like structure binary vocabulary

This kind of hierarchically based approach makes easier and faster the queries to the map keyframes when relocalization, after a tracking failure, is needed, or when loop detection is performed to understand if a particular place has been already visited and in this way correct the poses and feature points location linked to the keyframes in the database. In order to optimize 3D reconstruction and in this way camera pose estimation local bundle adjustment is performed on-line by solving a minimization problem of the reprojection error. As previously mentioned cost functions vary from implementation to implementation, but, in order to make an example, the solution proposed by Mur-Artal et al. in [60] is presented.

$$(\boldsymbol{R},\boldsymbol{t}) = \arg\min_{R,t} \sum_{i\in\chi} \rho\left(\left\|x_s^i - \pi_s\left(\boldsymbol{R}\boldsymbol{X}^i + \boldsymbol{t}\right)\right\|_\Sigma^2\right) \tag{2.48}$$

Where $x_s \in \mathcal{R}^2$ is the keypint, $X \in \mathcal{R}^3$ is the world point, ρ is the Huber cost function, π_s is the projection function obtained from triangulation. When this bundle adjustment is not performed on a small set of keyframe (local), but it is done for

example on the whole map after a loop detection, it takes the name of global bundle adjustment. In following figure 2.34 it is possible to see the life cycle of a mapping thread.

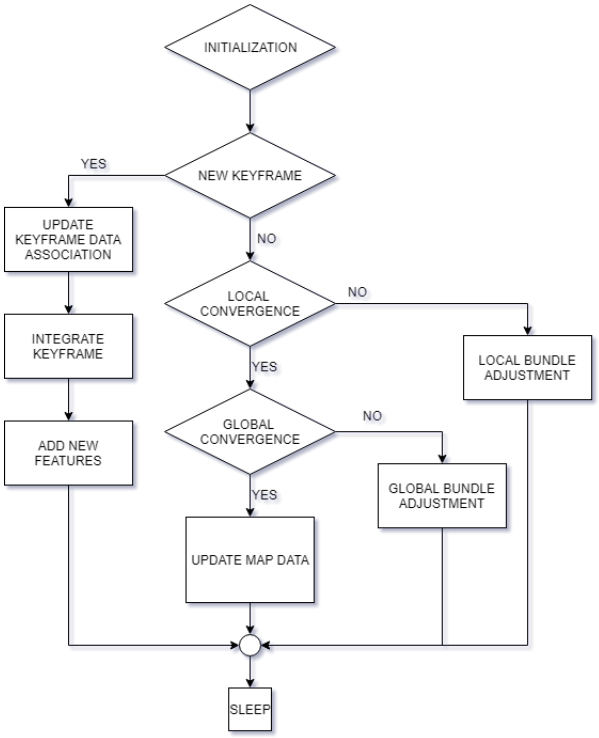


Figure 2.34: Qualitative flow chart of map building procedure

3 Innotech autonomous vehicle

As previously mentioned this paper aims at describing the work done for the start up Innotech Sys. The final objective was to design and build an autonomous robot able to escort and provide information and services to consumers of different realities. In particular, this project, has been carried on with the supervision of San Diego airport, California, indeed, the robot has been tested and optimized for airport indoor utilization. The driving idea of the project is to provide a powerful tool for the travelers in order to make the journey easier and more comfortable. Nowadays airports can be really challenging environments, and people not used to deal with them can encounter several difficulties in these places. Age, language barriers or simply impracticability are just some of the hindering factors this project wants to tackle. An example of the operational flow will better clarify the idea. The customer enters the airport and approaches the robot rest zone, scans his ticket into the robot and via vocal command asks to the AGV to escort him to the destination that can be a restaurant, the check in desk or the gate. The robot will start escorting the customer carrying his luggage to the final target.

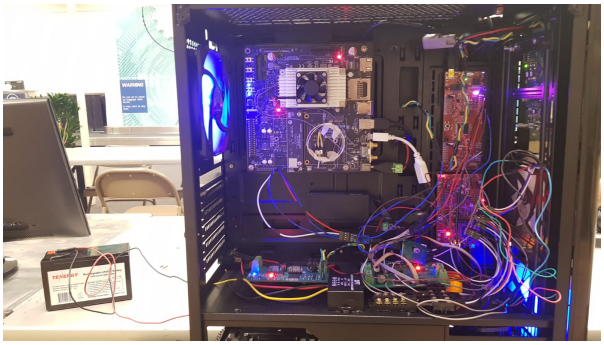


Figure 3.1: Side view of the prototype

3.1 Schematics

Before describing each part of the robot in detail it is convenient to provide a general description to have a better insight on how the system works and communicates under a macroscopic point of view. With reference to Figure 3.2 it is possible to see that

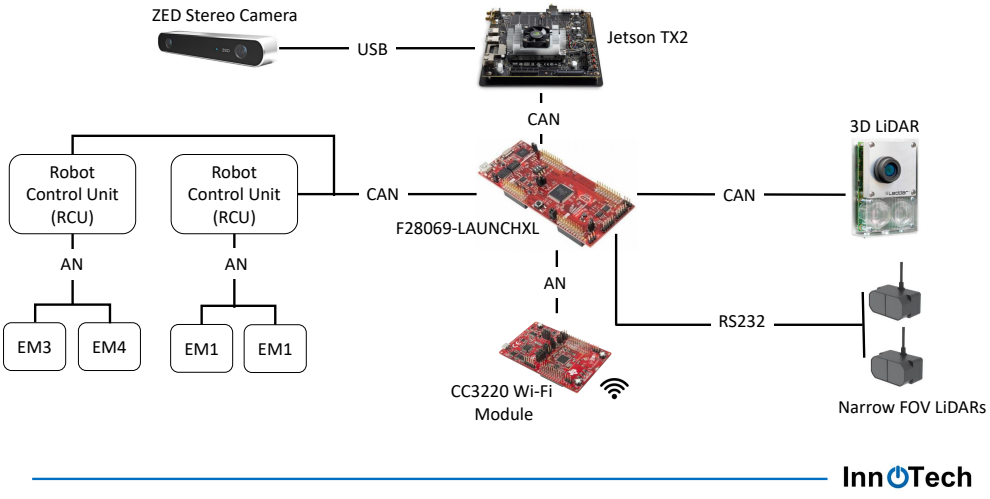


Figure 3.2: System Architecture

the key of the system is the Jetson TX2 where all the high level software is running, it is in charge of reading the different peripherals and process data for navigation, localization and object recognition and avoidance. The board communicates over USB with the ZED Stereo Camera that takes images of surrounding and rectifies them. Moreover it communicates with the main RCU F28069-LAUNCHXL via CAN bus. The RCU is in charge of interfacing the lidar with the Jetson and runs also the motors control algorithm. Motors commands are send via CAN bus from the main RCU to two smaller custom printed PCBs RCUs provided with H-bridges to power the motors. The latter work also as power distribution board in order to manage the battery packs. The design and development phase has been carried on a prototype AGV (Figures 3.1 and 3.3) in order to test what can be called the brain of the robot. This choice has been made in order to make all the hardware integration steps as smooth as possible carrying in parallel a mechanical design development updated with last changes and improvements.

Figure 3.3: Front view of the prototype

3.2 Datasheets

Having seen from a macroscopic point of view how the system has been designed it is now convenient to provide a detailed description of the principal hardware components. For the reasons already described no particular attention was paid to the mechanical side of the prototype, hence in this paragraph only the hardware relevant from a software point of view will be described.

3.2.1 ZED stereocamera by Stereolabs

ZED is a stereocamera developed by Stereolabs that has been used as main reality perceiver sensor. The choice has been driven by the quality provided from this sensor in image capturing and the strong partnership between Stereolabs and NVIDIA, hence providing optimal integration and working conditions of the hardware. Moreover, with the camera it comes its SDK that allows to build SLAM application characterized by massive speed and accuracy that will be described in following chapter 4. Following Tables from 3.1 to 3.5 report the technical hardware specifications.

Table 3.1: Video specifications

Video Mode	Frames per second	Output Resolution (side by side)
2.2K	15	4416x1242
1080p	30	3840x1080
720p	60	2560x720
WVGA	100	1344x376

Table 3.2: Depth specifications

Depth Resolution	Same as selected video
Depth Range	0.5-20 m (1.64 to 65 ft)
Depth Format	32-bits
Stereo Baseline	120 mm (4.7")

Table 3.3: Motion specifications

6-axis Pose Accuracy	+/- 1mm
	Orientation: 0.1°
Frequency	100 Hz

Table 3.4: Sensor specifications

Lens Field of View	90° (H) x 60° (V) x 110° (D)
Sensor Resolution	4M pixels per sensor with large 2-micron pixels
Dimensions	175x30x33 mm (6.89 x 1.18 x 1.3")
Weight	159 g (0.35 lb)

Table 3.5: Power specifications

Connector	USB 3.0 port with 1.5m integrated cable
Power	Power via USB 5V / 380mA
Operating Temperature	0°C to +45°C (32°F to 113°F)

3.2.2 Jetson TX2

The principal computer embedded in the system is the Jetson TX2 development board. On this hardware both the AI Yolo application for object recognition and SLAM system run. Following figure 3.4 and Table 3.6 show the technical specification and schematics. Detailed explanation of the software running on this hardware will be held in the dedicated Chapter 4.

Table 3.6: JetsonTX2

GPU	256-core NVIDIA Pascal^{TM} GPU architecture with 256 NVIDIA CUDA cores
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore
Memory	8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
Storage	32GB eMMC 5.1
Power	7.5W / 15W

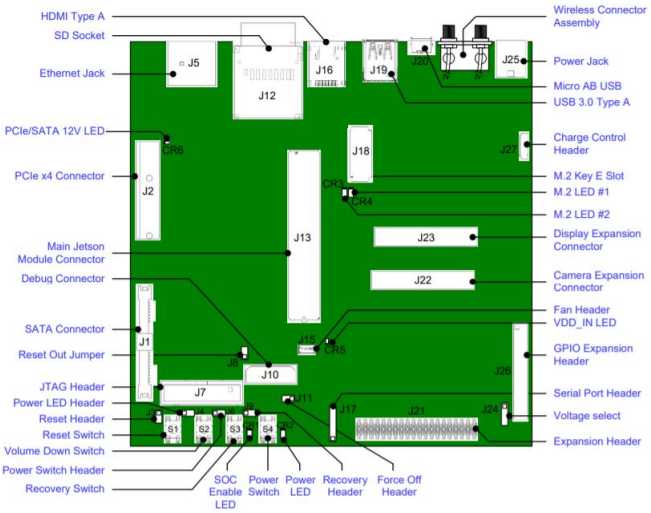


Figure 3.4: Jetson TX2 schematics

3.2.3 F28069-LAUNCHXL by Texas Instruments

This controller runs the control algorithm of the motors. The software has been entirely developed in Model Based Design in Simulink environment. The above men-

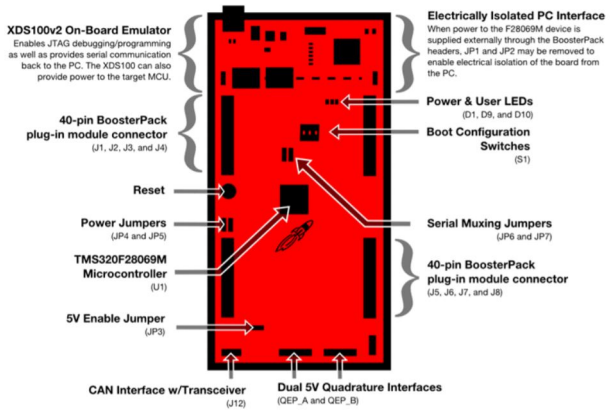


Figure 3.5

tioned software takes as input the CAN messages received from jetson TX2 in order to

consequently power the motors. No detailed description of the logic will follow since this would go out of the scope of the work. Anyway, in this section an insight about the integration of the stepper motor encoder will be given since this has been one of the responsibilities of the author. The entry point of the control block in Figure 3.6 is the eQEP. The enhanced quadrature encoder pulse (eQEP) block is used along with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine. In other words, this simulink block reads the information provided by the encoder in order to use them in the feedback of the stepper motor controller. The control logic is based on a negative feedback provided by the angular

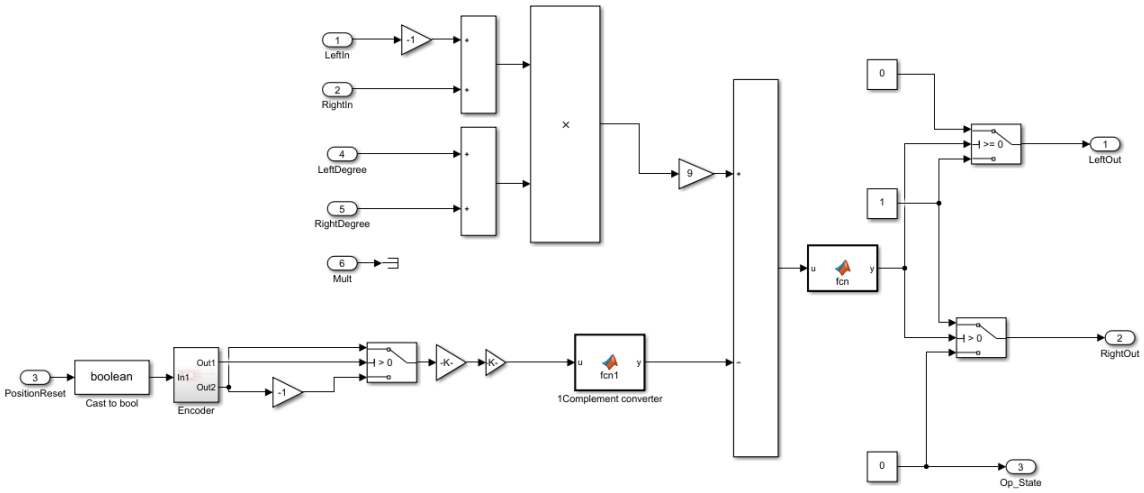


Figure 3.6: Encoder based control block

position of the stepper motor. A turn command in degrees is accepted via CAN and compared with the encoder output when this difference is bigger than a threshold the command to power the stepper motor is sent, while when the difference falls inside a certain range the motor is idle. This is done to avoid hysteresis cycles. Anyway, due to imperfect motor coupling and noise on the motor control board the stepper motor was still rotating too slow to assure good control and motion performances, this is why, in order to make operations smoother and more efficient also tangential velocities of the wheels have been exploited in order to create a differential aiding in the stirring phase. In Figure 3.7 it is possible to see the implemented logic. Through a basic logic circuit the angle direction of rotation is deduced, and according to it,

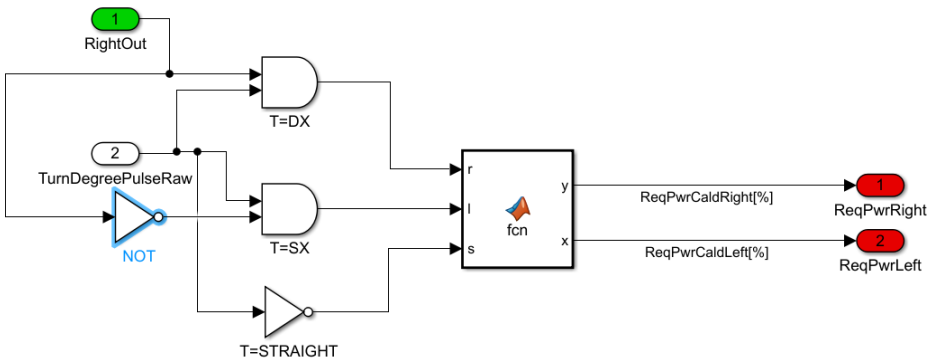


Figure 3.7: Differential aiding

wheels DC motor are powered differentially. In steady conditions motor are powered for 75% of the nominal power, in stirring phase the wheels nearer to the center of rotations are powered for 50% while other wheels are powered in steady conditions.

3.3 Adopted communication protocols

The designed AGV in order to accomplish its tasks needs to communicate both internally and externally to the system. Communication over different hardware composing the overall system is essential from basics operations, such as moving, stopping or stirring, to more advanced functionalities which can be detecting, recognizing or avoiding obstacles. On the other side the machine must be also able to communicate with external entities such as controllers, tablets or even other similar robots. To the aim of designing an efficient system different communication protocols must be taken into account considering strengths and weaknesses of each of them. The final choice has fallen on CAN bus for internal communication and MQTT network protocol for external information exchange.

3.3.1 CAN bus

Controller Area Network bus, or in short CAN bus, is a message based protocol able to make different microcontrollers communicate without the need of a host computer. It has been officially released in 1986 by Bosch[62]. CAN bus is a multi master serial

bus, this means that more than one node (each ECU is called node) can start the transmission since the hierarchy is not always vertical. Moreover, being serial, low lines are used and this allow a low pin count and reduced hardware cost. This protocol started to be used mainly in automotive application, but because of its efficiency spread in each field of automation and embedded systems. It is the De Facto standard for automotive nowadays and due to its lightweight protocol management, low cost and deterministic resolution of contentions it appears as an appealing solution for high speed and efficiency applications. Can physical layer is basically composed by two lines for data transmission CAN_H (CAN high) and CAN_L (CAN low). Their digital voltage levels determines if 1 or 0 is transmitted. The differential signaling ensures current flowing in conductors of equal intensity but opposite in sign, resulting in a field canceling effect carrying really low noise. Usually wires are twisted in pairs for a maximum length of 40 meters allowing a maximum of 30 connected nodes.

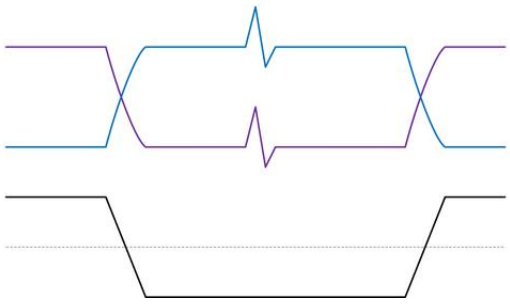


Figure 3.8: Differential signal example

As previously mentioned CAN bus has been used to implement the communication between the Jetson TX2 and the main RCU directly talking to the motors control boards. The information that the main computer sends are relative to the directions commands of motion. In particular the Jetson acquires the objective coordinates in the 3D space as cartesian reference and calculate the trajectory knowing its current position. The structure of a message sent over CAN bus is a standard form meant to ensure integrity and correctness of data as well as efficiency in trasmission. In particular this shape is well explained by Figure 3.9.

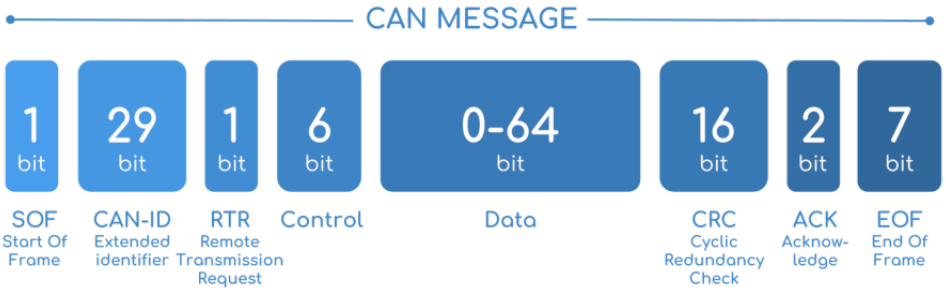


Figure 3.9: CAN bus message data structure

The first bit is called Start of Frame (SOF) and it is intended to communicate the will of a node to talk, furthermore it has the value of a *Dominant Zero*. SOF is followed by the CAN identifier(CAN-ID). This frame is important for two reasons: the first is that it uniquely identify a message, indeed providing useful information about the sender and kind of data transmitted, while the second is the priority of the message, lower values have higher priority. Next part of the structure is the Remote Transmission Frame (RTR), which even if little used, indicates whether a node is sending dataor asking fro specific messages. Following the RTR it is possible to find the Control field, also named Data Length Code (DLC). This last is made up of 6 bits indicating the length of the sent message data. After the DLC there is the core of the CAN message shape, the Data field. Data is sent over CAN in a unreadable way, that is why information must be decoded in order to be visualized in engineering values or human-readable form. Once data are transmitted a quality check is demanded to the Cyclic Redundancy Check frame (CRC), which ensure integrity of information. The structure of the message closes with 2 bits for the Acknowledgment (ACK) with which receiver nodes provide confirmation of the received data and 7 bits for the End of Frame (EOF). Without entering in the details of the bit timing it must be said that every node needs a CPU, a CAN controller and a CAN transceiver in order to exploit this communication protocol. The way chosen in order to structure the data field has been a 8-Bytes bitmask.

Bytes number 1, 2, 3 and 5 corresponds to a direction command and all of them

Figure 3.10: CAN data bitmask

power the wheels motor, in particular all of them will power the motor in the positive direction while the second one will power them in the opposite direction. Bytes number 4 and 5 can range from 0 to 4 in function of the amount of stirring requested. The stirring of the robot is actuated by a stepper motor which rotates the platform on which the front semiaxis is mounted. The value 0 corresponds to the unstirred direction of motion while values from 1 to 4 are discrete values corresponding to incremental rotations of the stepper motor. Byte number 7 is the STOP command which is given when the final objective is reached or a condition preventing the robot from continuing the mission happens. Last but not least byte number 8 is unused in normal conditions and indicates a generic error situation adopted to put the robot in safety condition when an unknown and not controlled event happens.

3.3.2 MQTT

Message Queue Telemetry Transport, or in short MQTT, is a message based protocol typical of application where power consumption is a critical point and resources from this point of view are limited. Anyway, this protocol results to be so pragmatic and efficient that even Facebook Messenger app decided to base its working on MQTT, asserting that the latency in sending/receiving messages reduced from multiple seconds to hundreds of milliseconds, meaning a speed increase of one order of magnitude. This protocol appeared to be fitting perfectly with project that needed to interface the robot with external entities, which could be other similar machine, workstations or related mobile applications. The above mentioned messaging protocol does not link directly the talker (now on indicated as publisher) and the listener (now on indicated as subscriber) but interposes between them a third entity: the broker. The broker is

the only server while publishers and subscribers are clients. Brokers can be whether installed on the same machine or be external services in the same network or even outside it. The protocol works in a simple fashion: publishers publish messages on a particular topic, intended as a queue of messages, and subscribers receive messages coming from the topics they subscribed to; brokers are in charge of receiving messages from publishers and redirect them to correct subscribers. The structure of a MQTT message can be described by following Figure 3.11.

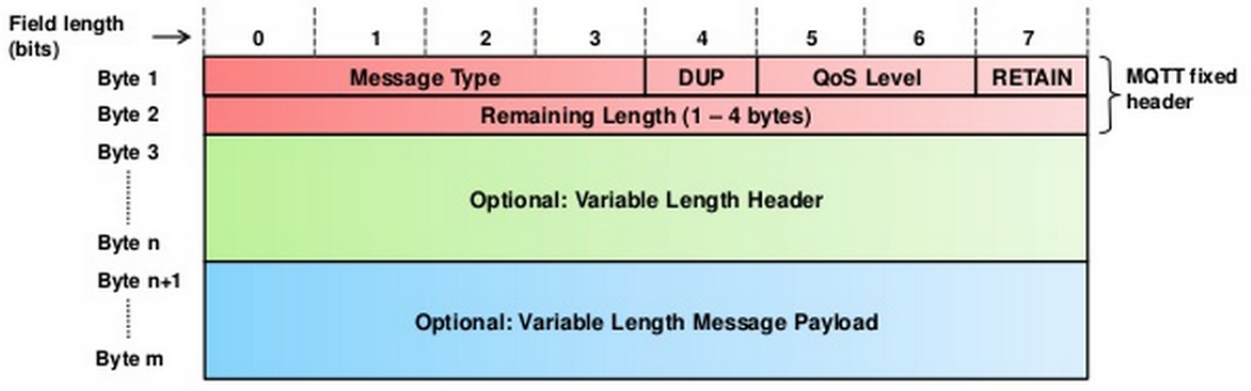


Figure 3.11: mqtt message structure

The structure is basically divided in two parts, header and payload. The aim of the header is to describe the payload and make sure clients and broker have enough information to handle the message. Header itself is divided again in two parts, fixed and variable length section. The MQTT fixed header is the part where following information are provided:

- *Message type*: this section indicate to which protocol command or response the message belongs. It can be a subscription, publication, connection and so on.
- *Duplicate Message flag(DUP)*: this field inform the subscriber that the message could be already been delivered.
- *Quality of Service (QoS) Level*: this is a delivery assurance on the published messages. It can range from 0 to 2 and indicates respectively: at most once the

message is delivered, at least once the message is delivered and exactly once the message is delivered. In case of mismatches of QoS between different clients lower level is effectively respected.

- *Retain*: This is flag that suggests the server to deliver the last published message at the first subscription.

- *Remaining length*: This field indicates the number of bytes still present in the rest of the message, i.e. the two optional frames of the variable header and payload.

The variable section of the header is function of the kind of message taken into consideration, just to make an example, in case of a message of type PUBLISH the information stored inside this field can be the topic. Last but not least there is the payload section, which has not a prescribed length, and it is the part where the effective information is stored. Coming to the way MQTT has been implemented into the developed AGV, this protocol is the way the robot gets the commands from the developed android app acted at interfacing the service with the customer. Furthermore, the protocol is also exploited to inform the user interface (UI) about robot state and position. Examples of data exchange are the directions the robot is following, the hand shake for mission start or completion and eventual lost signal. Moreover, when the robot is commanded in manual mode the joystick always implemented in an android app sends the directions commands via MQTT. More details about this matter will be given in Section 3.5.

3.4 Shape design

Different concepts have been evaluated regarding the chassis of the robot. Each of them has some peculiarities and distinctions but they are united by two main characteristics: all of them have the hardware and battery packs at the bottom of the body and all of them have a screen on the top in order to be visible and of easy use by the customer. Two similar shapes are shorter and with a smaller screen in order to

make room for storage in the middle. The three remaining shapes are, on the other side, united by a more long limbed design acted at presenting a bigger screen of even easier and more comfortable use for the costumer.

3.5 User Interface (UI)

Nowadays robots are becoming part of people everyday life, anyway there is still some kind of distrust towards these kind of technologies in domestic or commercial application. This is the main reason why the way people interface themselves with the AGV must be easy, efficient, rapid and effective. A mobile application-like design is the synthesis of these pillars.

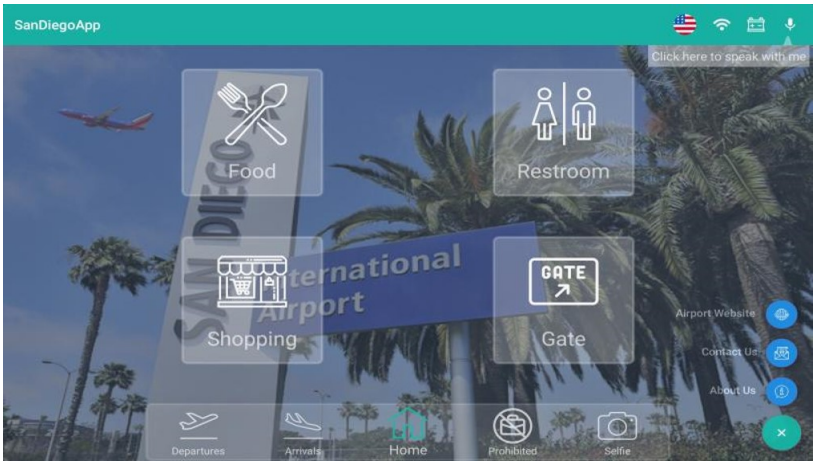


Figure 3.12: User interface homepage

The homepage visible in Figure 3.14 is what a costumer would see on the screen on top of the robot. From this main page he could navigate inside the application finding different useful information, such as the map of the airport with the restaurants, Figure 3.15.

The main activity that the user can exploit is the navigation system which will escort the costumer to the selected interest point. This service, as anyone else provided by the application, can be selected with the touch screen or by vocal command. The entry point is a list of the interest points, Figure 3.16

Each of the item in the list is uniquely linked to a three-dimensional coordinate on

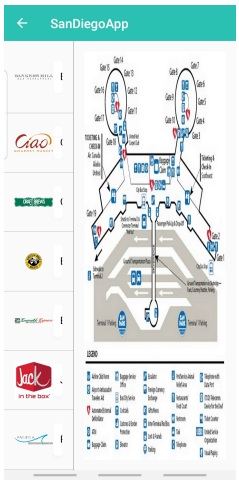


Figure 3.13: Airport map

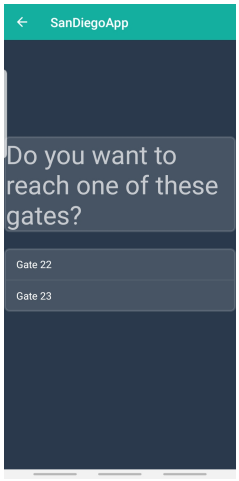


Figure 3.14: Interest points list

the map of the robot stored in the SLAM system, once the person selects the final destination, this will be communicated to the SLAM system via MQTT. After the handshake between interface and robot happens, the application will ensure the readiness of the costumer to follow the AGV and eventually this last will start moving. During the mission directions and state of the robot are provided to the person via screen in a user friendly fashion. All data is exchanged via MQTT, again. Last but not least, five different languages are supported and menu, as well as vocal assistant, will be set to the preferred language, Figure 3.17.

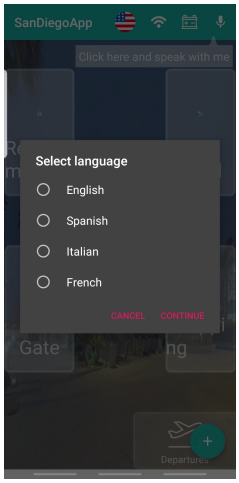


Figure 3.15: Language selection

4 Robot Localization and Mapping

The scope of this chapter is to provide an insight on the principal work done by the author, that is the implementation of the Simultaneous Localization and Mapping system on the AGV. After the review of the literature of the matter a first choice was to be made: visual or lidar SLAM implementation. Both solutions were possible and each of them had advantages and drawbacks. The final decision has been to opt for the visual. The reason behind this choice has been the presence of the ZED stereo-camera, already integrated because of YOLO object recognition software. Including a new sensor, such as lidar, would have meant increased costs and both hardware and software complexity in the final product. The cost, indeed, has been one of the driving factors in the process of design, since the service robot target was a low-cost application. Description of the developed code will follow, comparing two different libraries used to implement the SLAM system, the proprietary ZED SDK and the open source orbSLAM2. Tests and results will be also described.

4.1 Implemented SLAM system

As previously mentioned two different libraries have been exploited in order to build the software, anyway the principal work flowchart of both the systems is similar. When the application is started five different command lines options must be passed to the software, which will execute different branches in function of user preferences.

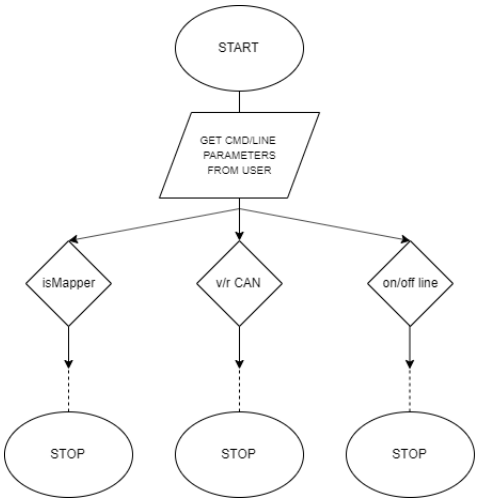


Figure 4.1: Ramification of the workflow

Of the five different command lines parameters already mentioned only three of them individuate separate flows (Figure 4.1). These parameters are:

- *isMapper* is a boolean value indicating whether the application must be started in mapping or driving mode.
- *virtual/real CAN* is a boolean value indicating whether the application should use the virtual or real CAN interface.
- *on/off line* is an optional parameter, which if present will instruct not to read Stereocamera images but instead charging a pre-recorder stereovideo or image sequence.
- *turning and position tolerances* are the two remaining parameters that does not interact directly with the execution of the program, but just indicates the tolerance threshold to be respected in navigation phase.

4.1.1 Virtual/Real CAN bus

One of the settings parameters passed through command line is the `socket_TF` boolean variable. This variable must be set to 1 if a CAN physical interface is present, while can be set to 0 for testing purposes and the software will send virtual CAN messages exploiting the library SocketCAN. SocketCAN is a set of drivers developed by Volkswagen in order to implement CAN protocols for Linux. There are several CAN implementations on Linux based machines, but one the particularities of SocketCAN is that it has been created in a network programming fashion that is meant to be as close as possible to TCP/IP protocols. This choice was made to allow programmers to become familiar with CAN sockets in the smaller time possible. Furthermore, excluding SocketCAN, these kind of CAN implementation are usually hardware specific with a comparatively little functionality. Exchanging the CAN controller would mean an update in the software leading to adaptation of large parts of the application. SocketCAN being not implemented in the user space, grant to the programmer a socket API that can be called and programmed without taking care of the specific hardware.
Moreover, SocketCAN has a useful functionality that is the virtual CAN socket. The exactly same code can be used to instruct a real CAN network or a fake one aimed at testing purposes. The only difference between the two implementations is the kind of socket to be used.

4.1.2 Mapping

The mapping mode is the one used to map a new environment, hence no motor output commands will be given.
Mapping application has been usually exploited in off line mode, since on line mapping would result in lower quality and, anyway, it was not required by the application. This is why, during the mapping the robot was commanded in manual mode, a stereovideo of the environment was recorded and in a second time analyzed. To this aim an on purpose software was coded and interfaced with a joystick android application.

The manual mode works in a simple manner. It initialize CAN and MQTT interfaces

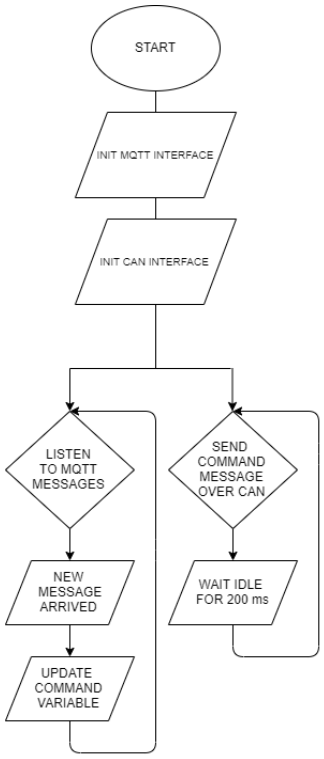


Figure 4.2: Manual control flowchart

and starts two different and separated loops, one for CAN bus and one for MQTT subscriber. The two loops run at different frequencies, in particular 5 Hz is the frequency chosen for the CAN messages, while the subscriber continuously listens on the MQTT network waiting for messages published by the publishers, which in this case is the android joystick application. In order to choice the broker both the external and internal ways have been considered. At first place and external broker was exploited, but the latency for the message queue was to high and the joystick was not responding enough fast to the inputs. This is why, in second place it has been decided to install a broker in the internal network, reaching sufficiently low response time.

4.1.3 Navigation

Before diving into the life cycle of the main branch it can be useful to provide an overview on the classes defined into the software. Three different classes are reserved to communication protocols interfaces, one for CAN bus and two for MQTT. The class `CAN_interface` is defined as follows:

```
class CAN_interface{
    int s;
    struct can_frame frame;
private:
    void sendOK(void);
public:
    int CAN_send(int byte);
    CAN_interface (bool socket_TF);
    void CAN_loop(void);
};
```

The constructor will initialize the proper socket according to the already discussed boolean variable `socket_TF`, prepare the structure of the CAN message and start a detached thread `CAN_loop`. `CAN_loop` is a message sender cycle that reads the value of the shared variable `LAST_COMMAND`, convert its value to the proper CAN message and send the message over the BUS to the RCU. Another important method has been implemented in order to fulfill an handshake with the rest of the system at the boot, this method just send a void message correctly formatted once. The last method is the `CAN_send`, which has been implemented just to send a single CAN message in case of unscheduled events. To complete the description about communication protocols other two classes must be presented, `MQTT_interface` and `subscriber`. Both of them implemented MQTT interfaces but they are separated since they start two different threads detached from the main application.

The definition of the `subscriber` class is the standard reported in the official github repository of Eclipse Paho project [63], hence not reported. The only modifications

were made to adapt the topic and the callback `on_message`. The callback, when called, calls in turn a function that process the message event, for example starting or aborting the mission. The other MQTT based class is the `MQTT_interface`, of which definition follows:

```
class MQTT_interface{
public:
    mqtt::async_client pub{SERVER_ADDRESS,PUB_ID};
    MQTT_interface ();
    void pub_loop(void);
};
```

It is possible to see at first place the declaration of the client `pub` which define the instance of the publisher object. Following, the constructor, similarly to `CAN_interface`, set up the client and detach an independent thread, `pub_loop` for publication which publishes a message with a frequency of 5 Hz. Going deeper into the core of the application it is possible to find the class `Commands`, in charge of taking the information from the SLAM system, elaborate them and communicate formatted data to the communication protocols classes.

```
class Commands{
    int cnt,heading;
    float positional_tol,turning_tol;
private:
    int isnotRange(float a,float b, float t);
public:
    Commands(float pos_tol, float turn_tol);
    int Calculate_and_command(Rotation Rwc,Translationt wc,
                                        ,vector<vector<float>> &ref);
    vector<vector<float>> get_ref_dir();
    int processKeyEvent(Translation twc,vector<vector<float>> &ref);
};
```

From the declaration it is possible to see the constructor `Commands` that initialize some settings parameters that must be tuned in order to improve the quality of the navigation, example of these can be the tolerance parameters passed via command line already discussed prior. Before introducing the most important methods of the class it is useful to discus `isnotRange`. This methods is exploited constantly during navigation phase, it is a simple function that returns boolean values. Its aim is to tell whether a value `a` falls inside a linear interval of radius `t` centered in `b`.

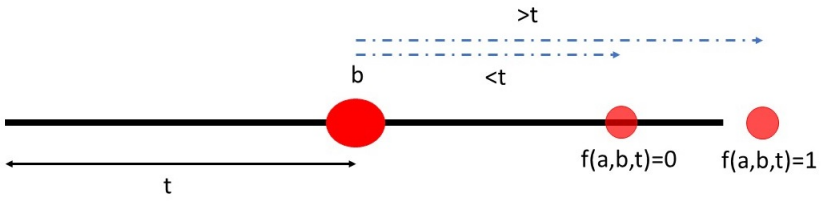


Figure 4.3: isnotRange function

The trajectory, being shaped by a set of subsequent waypoints, forces the robot to follow an imaginary trackline, the `isnotRange` method is constantly adopted to understand if a certain waypoint is reached and in case step to the next one. Having cleared how the trajectory is followed it is possible to introduce how it is loaded into the program. The first step is upon the tablet application, which asks the user which is the final objective. This is sent to the path planning algorithm that look for that place in a file, linking each interest point to a three-dimensional coordinate, and create a .txt file with all the waypoints. Once this procedure is finished the SLAM application can call the method `get_ref_dir` that will search for the above introduced .txt file and load those waypoints in the memory as a vector of vector. Finally the last two methods are `processKeyEvent` and `Calculate_and_command`. The first is a debugging and security escape function. It can be called by remote in order to print on the screen some debugging variables such as: position, orientation, MQTT messages queues, CAN bus messages etc. It can also be used in order to cancel the mission or stop the robot if something unexpected happens.

Last but not least `Calculate_and_command` is the piloting method. It gets as inputs

orientation and position of the vehicle as well as waypoints, check if the current position is near one of them and if it is case load in the memory the next waypoint, in the opposite case it will drive the robot in the correct direction by sending the right CAN message. It also control the stirring since when a curve must be tackled it will command the stepper motor to rotate until the requested degree is reached and will communicate the motor when to come back to the straight position. The last functionality implemented into this methods is the drifting correction, in fact, the robot is instructed to follow a straight imaginary line during rectilinear. In order to do this the stepper motor is commanded to stir whenever the AGV runs further than a tunable threshold from the reference line.

Having explained each part of the implemented software is now possible to introduce the general flowchart of the life cycle of the application, from the directions instructions to the reach of the final objective (Figure 4.4).

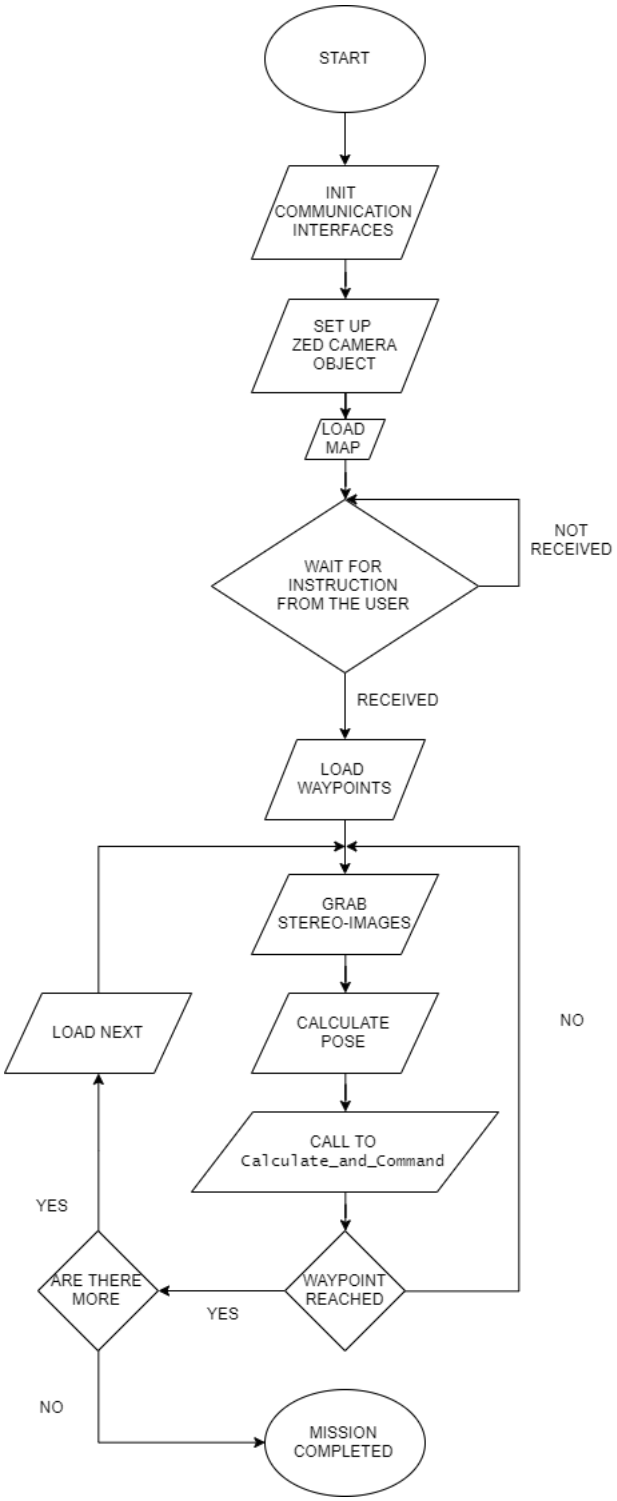


Figure 4.4: Life cycle flow chart

4.2 Testing

The tests to be performed have been indicated by the San Diego airport, indeed they asked to perform at least 30 successful missions in order to validate and evaluate the project. The tests were organized in the following way.
A potential user, in that case airport employees, would use the graphical interface to ask the robot to bring him or her to one of two possible gates, the robot would assist the user to the destination and come back to the original position. Two possible paths were available, one straight line an done L-shaped curve.

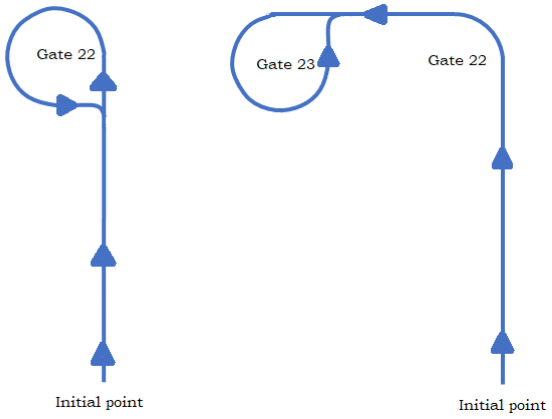


Figure 4.5: Straight path on the left and L-shaped on the right

To the aim of evaluating the quality of the developed application some preliminary tests have been performed inside the Engineering and Technology department of California State University of Los Angeles. These were focused on understanding how precisely can a vehicle localize itself under a mapped environment. The software developed for SLAM purposes exploits the above cited ZED SDK or orbSLAM2 in order to build a map of the environment and then localize into it. A ZED stereo-camera has been used to record two different videos of a L-shaped path inside the department. Each sequence has been analyzed once to build a map and, in a second phase, two different localization tests have been performed on each generated map by feeding the system with a new stereovideo of the same L-shaped path. It is worth to notice that no true information about trajectory were available during the tests, and

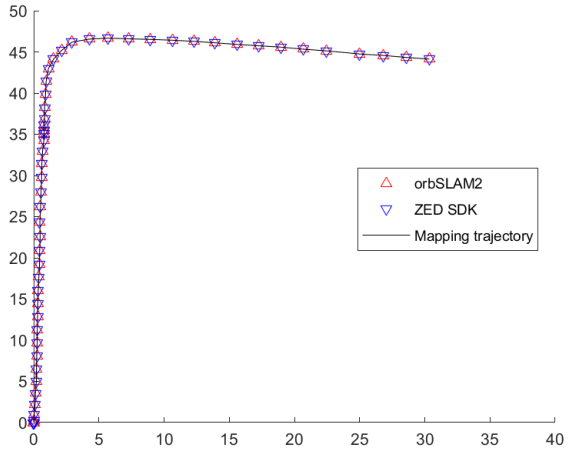


Figure 4.6: Sequence 1 trajectory of the L-shaped department path

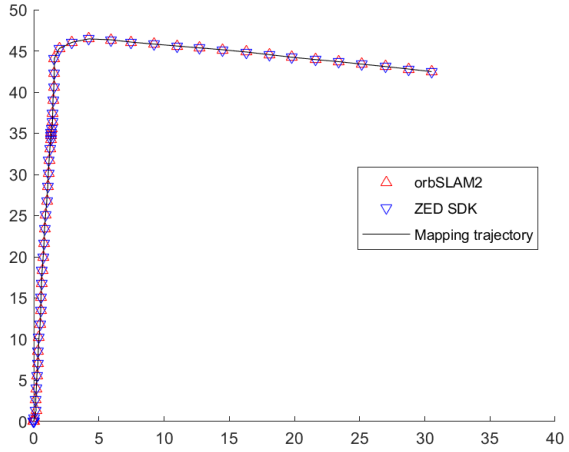


Figure 4.7: Sequence 2 trajectory of the L-shaped department path

because of this the vector of camera poses recorded during the mapping phase has been assumed as true estimation and compared to the measurements taken during the evaluation attempts. Must be pointed out that this assumption does not lead to a loss of generality since, even if the map reference frame is not coincident with the world reference frame, the localization technique would still serve the purpose of localizing inside the mapped environment. Landmarks and features are stored inside the map and they are assumed to be still in space, indeed a relative localization with reference to them would be possible.Results of tests are reported in Table 4.1.

Table 4.1: Tests Results

	N of frames	Mapping time	mean FPS		$\mu_{err_{rel_x}}$	$\mu_{err_{rel_z}}$	μ_{err_x}	μ_{err_z}	σ_x^2	σ_z^2	Number of track losts
Sequence 1	982										
		270 secs	3.6	*orbSLAM2*	2.022%	1.53%	0.6 mm	1.6 mm	0.00013	0.00019	0
		30 secs	30	*ZED SDK*	2.21%	1.17%	0.7 mm	1.7 mm	0.00012	0.0002	0
Sequence 2	1039										
		306 secs	3.4	*orbSLAM2*	0.67%	1.56%	0.8 mm	2.2 mm	0.00008	0.00017	0
		34 secs	30	*ZED SDK*	2.32%	2.11%	1.2 mm	1.3 mm	0.00009	0.00017	0

From the measurements of variance it is possible to notice the repeatability of the localization phase permits reliable operations since it remains fairly constant among different attempts. As far as mean localization error is concerned values are in the order of millimeters; it must be anyway pointed out that this error does not represents the absolute position estimate value but just the noise to take into account when referring to a previously built map.

As it can be derived from measurements reported in Table 4.1 the two different libraries do not behave consistently in a different way, in fact the results of variance and tolerance are absolutely comparable, indeed almost equal. What differ between the two different methods is the massive speed characterizing the ZED SDK, which makes possible to run the SLAM system at 30 FPS, while orbSLAM2 remains on an average value of 3.4-3.6 frames per second. It has not been possible to investigate what makes this huge difference since, when contacted, Stereolabs did not want to share any information on its proprietary software. It must be anyway pointed out that the increased FPS at which the application can run when deployed with ZED SDK bring two big advantages: reliability of the localization and mapping in increased and the possibility of lost track is way reduced due to the accurate features tracking derived from the high frequency, the second advantage is the possibility of running the software in odometric way that means with no need of a previously charged map. This last situation comes with some drawbacks that are the loss of reliability and repeatability in localization as well as the risk of loosing the track with no possibilities of getting it back. This was anyway not possible with orbSLAM2 due to the low FPS, and, even if this does not represent a good practice for SLAM purposes, still presents an added feature.

Having tested and validated the implemented system three days have been spent in the airport innovation lab, which is a dismissed terminal, in order to collect feedback and surveys from customers. Almost 50 tests have been performed and the obtained feedback were promising, Figure 4.8.

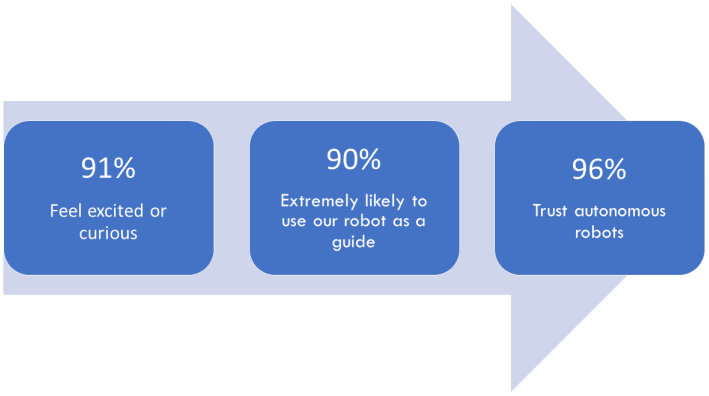


Figure 4.8: Results of survey

5 Conclusions and future lines

Coming to the conclusions it is possible to sum up what the work was aimed to. In this paper an extensive description of Simultaneous Localization and Mapping state of the art has been performed as well as a comparison of the most recent image processing and description algorithms. A possible implementation of these technologies in the service robotics field has been investigated and tested through the designed prototype of steward robot.

Thanks to the California State University, San Diego airport and Innotech-sys the prototype has been built and tested in a real airport environment providing useful insight on possible diverse implementations and improvement of the project.

The tested hardware, and in particular NVIDIA Jetson TX2 and ZED stereocamera, behaved as expected from the performance point of view and confirmed what most of the implementations nowadays present in the research are experiencing. This hardware is able, thanks to its computational power and efficiency, to run in real time applications that are strongly limited by low power consumption still asking for sufficiently high performances. This is due to the massive parallel GPU computing.

Possible future improvements of the work done are various and diverse. The study carried on is just at an embryonic level. At first place a proper path planning service must be integrated in the system which is prepared to move in a unstructured environment but the lack of a proper trajectory planner strongly limits the potentialities of the AGV. Moreover, at the actual state the robot is trained to react to unexpected events and objects by stopping the motion, reaching a safe condition and wait for restoration of previous conditions. This means that if an obstacle comes through the robot will detect it and stop until the path is free again. This behavior can be

certainly improved by updating the trajectory in real time or dynamically react to moving objects.

Other possible future improvements include for sure the feasibility study of a lidar implementation. Lidar is a promising technology, widely used in this field; its cost is higher than a stereocamera implementation but final results could be more precise and easily portable. Last but not least the possibility of a steward robot fleet must be taken into account and robots must be made able to communicate with each other. This would result in sharing resources and useful information compensating the increased complexity with efficiency increase.

Bibliography

[1] SaviokeTeam. (accessed: 19.06.2019). [Online]. Available: savioke.com

[2] AethonTeam. (accessed: 19.06.2019). [Online]. Available: aethon.com

[3] Robotiq®. (accessed: 26.06.2019). [Online]. Available: robotiq.com

[4] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics and Automation Magazine*, 2012.

[5] Flir®. (accessed: 28.06.2019). [Online]. Available: flir.com

[6] Basler®. (accessed: 28.06.2019). [Online]. Available: baslerweb.com

[7] L. Li, "Time-of-flight camera – an introduction," *Technical white paper for texas instrument*, 2014.

[8] STEREOLABS. (accessed: 28.06.2019). [Online]. Available: stereolabs.com/zed

[9] NaionalInstruments®. (accessed: 01.07.2019). [Online]. Available: ni.com

[10] M. Santoro, G. AlRegib, and Y. Altunbasak, "Misalignment correction for depth estimation using stereoscopic 3-d cameras," *School of Electrical and Computer Engineering, Georgia Institute of Technology*, 2012.

[11] F. Boisset. (2018) The history of industrial automation in manufacturing. (accessed: 18.06.2019). [Online]. Available: kingstar.com/the-history-of-industrial-automation-in-manufacturing

[12] B. Blue. (2013) Advantages and disadvantages of automation in manufacturing. (accessed: 18.06.2019). [Online]. Available: vista-industrial.com/blog/advantages-and-disadvantages-of-automation-in-manufacturing

[13] C. Mandel, K. Huebner, and T. Vierhuff, "Towards an autonomous wheelchair: Cognitive aspects in service robotics," 2005.

[14] RBRstaff. (2015) Savioke relay autonomous delivery robot. (accessed: 19.06.2019). [Online]. Available: roboticsbusinessreview.com/consumer/relay/

[15] KoreaBizwire. (2018) Incheon airport introduces "airstar," passenger aiding robot. (accessed: 19.06.2019). [Online]. Available: http://koreabizwire.com/incheon-airport-introduces-airstar-passenger-aiding-robot/121298?ckattempt=1

[16] DeutscheBank, "Breaking through the noise on amzn,usps and fdx/ups," 2018.

[17] T. Mai. (2017) Global positioning system history. (accessed: 21.06.2019). [Online]. Available: nasa.gov

[18] U.S.Government, "Global positioning system standard positioning service performance standard," *4^{th} Edition*, 2008.

[19] P. Goeland, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," *Department of Computer Science Institute for Robotics and Intelligent Systems University of Southern California*, 1999.

[20] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Department of Mathematics and Computer Science,University of the Balearic Islands,Palma de Mallorca, Spain*, 2008.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once:unified, real-time object detection," *University of Washington, Allen Institute for AI, Facebook AI Research*, 2016.

[22] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," *International Conference on Robotics and Automation*, 2016.

[23] B. Schwarz, "Mapping the world in 3d," *Nature Photonics volume 4*, 2010.

[24] M. Naveed, D.Fofi, and S. Ainouz, "Current state of the art of vision based slam," *Universitè de Bourgogne, France*, 2016.

[25] P. Li, D. Farin, R. K. Gunnewiek, and P. H. N. de With, "On creating depth maps from monoscopic video using structure from motion," *Eindhoven Univ. of Technology,Philips Research Eindhoven,LogicaCMG Nederland B.V.*, 2006.

[26] Q. Wei, J. Shang, Z. Chao, and Z. Zhenpu, "A real-time 2d to 3d video conversion algorithm based on image shear transformation," *School of Computer Science and Technology, Changchun University of Science and Technology,Changchun, China*, 2015.

[27] P. Jensfelt and alters, "A framework for vision based bearing ony 3d slam," *International Conference on Robotics and Automation, Orlando, Florida*, 2006.

[28] E. Mouragnon and alters, "Monocular vision based slam for mobile robots," *International conference on Pattern Recognition, Hong Kong,2006*, 2006.

[29] L. Goncalves and alters, "A visual front-end for simultaneous localization and mapping," *International Conference on Robotics and Automation, Barcelona, Spain*, 2005.

[30] M. Milford and alters, "Featureless vehicle-based visual slam with a cosumer camera," *Australasian Conference on Robotics and Automation, Brisbane, Australia*, 2007.

[31] T. Lemaire and alters, "Monocular-vision based slam using line segment," *International Conference on Robotics and Automation, Rome, Italy*, 2007.

[32] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[33] V. Castaneda, D. Mateus, and N. Navab, "Slam combining tof and high-resolution cameras," *IEEE Workshop on Applications of Computer Vision (WACV)*, 2011.

[34] D. B. Gennery, "Stereo-camera calibration," *Image Understanding Workshop, pp 107-108*, 1979.

[35] W. Zhao and N. Nandhakumar, "Effects of camera alignment errors on stereoscopic depth estimates," *Machine Vision Laboratory, Dept of Electrical Engineering University of Virginia*, 1996.

[36] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol 14, No 10*, 1992.

[37] OpenCV. Camera calibration and 3d reconstruction. (accessed: 02.07.2019). [Online]. Available: docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[38] MathWorks®. Stereo camera calibator app. (accessed: 02.07.2019). [Online]. Available: mathworks.com/help/vision

[39] R. Owens. (1997) Computer vision it412: Epipolar geometry.

[40] M. Mann. (2004) Stereo camera calibration. (accessed: 09.07.2019). [Online]. Available: https://cs.nyu.edu/courses/fall14/CSCI-GA.2271-001/06StereoCameraCalibration.pdf

[41] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. chapter 9, pp 239-262, Cambridge University Press, 2004.

[42] ——, *Multiple View Geometry in Computer Vision*, 2nd ed. chapter 11, pp 279-283, Cambridge University Press, 2004.

[43] Z. Chen, C. Wu, and L. Tang, "Image rectification based on minimal epipolar distortion," *Proc SPIE*, 2001.

[44] D. Papadimitriou and T. J. Dennis, ""epipolar line estimation and rectification for stereo image pairs," *IEEE transaction on Image Processing, vol.5, no.4, pp 672-677*, 1996.

[45] S. Roy, J. Meunier, and I. Cox, *Cylindrical rectification to minimize epipolar distortion*, 1997, pp. 393–399.

[46] R. I. Hartley, "Theory and practice of projective rectification," *International Journal of Computer Vision*, 1999.

[47] A. D. H. Strasdat, J.M.M Montiel, "Visual slam: Why filter?" *Image and Vision Computing, Volume 30, Issue 2, pp 65-77*, 2012.

[48] M. M. G. Wyeth, "Featureless vehicle-based visual slam with a consumer camera," *Australian Conference on Robotics and Automation, Brisbane, Australia*, 2007.

[49] D. Tyagi. (2019) Introduction to feature detection and matching: Fast, harris, sift,surf,brief and orb. (accessed: 11.07.2019). [Online]. Available: https://medium.com/software-incubator

[50] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," pp. 430–443, 2006.

[51] C. Harris and M. Stephens, "A combined corner and edge detector," 4^{th} *Alvey Vision Conference*, 1988.

[52] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[53] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008, similarity Matching in Computer Vision and Multimedia.

[54] E. Rublee, V. Rabaud, K. Konolige, and G. Bradsky, "Orb: an efficient alternative to sift or surf," *Willow Garage,Menlo Park, California*, 2011.

[55] X. X. Lu, "A review of solutions for perspective-n-point problem in camera pose estimation," *Journal of Physics: Conference series 1087*, 2018.

[56] I. Gordon and D. Lowe, "Scene modelling, recognition and tracking with invariant image features," *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2005.

[57] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspace," pp. 225–234, 2007.

[58] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization," *ETH Zurich Research Collection*, 2013.

[59] Z. Dong, G. Zhang, J. Jia, and H. Bao, "Keyframe-based real-time camera tracking," *IEEE 12th International Conference on Computer Vision (ICCV)*, 2009.

[60] J. M. M. Mur-Artal Raúl Montie and T. J. D., "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, 2015.

[61] J. Engel, T. Schops, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," *Lecture Notes in Computer Science, 834-849*, 2014.

[62] M. DiNatale, "Scheduling the can bus with earliest deadline techniques," 2000.

[63] Eclipse. (accessed: 20.10.2019). [Online]. Available: https://www.eclipse.org/paho/clients/cpp/

[64] R. Mur-Artal and J. D. Tardos, "Fast relocalisation and loop closing in keyframe-based slam," *IEEE International Conference on Robotics and Automation*, 2014.