Università degli Studi di Padova

Department of Information Engineering

Master's Degree in Automation Engineering

INDOOR AND OUTDOOR LOCALIZATION FOR AGVs IN THE PRIMARY ALUMINUM INDUSTRY

Supervisor
Angelo Cenedese

Co-supervisors
Eng. Mauro Schiavo
Dr. Nicola Lissandrini

Master's candidate
Matteo Giacomazzo
1179770

Academic Year 2019/2020

“Life is a sum of all your choices” - Albert Camus

Abstract

This work is one of three theses realized thanks to the collaboration between the University of Padova and the Techmo Car S.P.A. company. Techmo is a world leader company in the production of high-end customized mobile and stationary equipment and operative vehicles for the primary aluminium industry.
The goal of this project is to provide a proof of concepts of the use of Autonomous driving in the primary aluminum industry. In particular this thesis reports the analysis of two types of indoor and outdoor localization techniques in an aluminum smelter with a particular operative vehicle, the *Fluoride Feeder*.
The indoor localization is performed with the ARTag Fiducial 2D Markers System: several planar tags placed in the environment are detected with visual cameras, using suitable Open CV and errors correction algorithms to retrieve the marker ID and its pose with respect to the camera optical axis.
The outdoor localization employs a Wireless Sensors Network (WSN) in which multiple Wi-Fi transceivers are adopted to trilaterate the position of the vehicle based on the Received Signal Strength Indicator (RSSI) value.
The raw estimated pose obtained using both the localization approaches is, then, fuse with the information retrieved by an ideal Inertial Measurement Unit (IMU) sensor using the Extended Kalman filter in order to increase the localization performances.
Experimental tests have been carried out in the Gazebo virtual environment using the Robot Operative System (ROS) and the Unified Robot Description Format (URDF) XML language to setup and initialize the simulations. Stationary and moving tests have been performed to deeply analyze the localization techniques behaviour in many possible conditions: in particular, in the moving simulations the fluoride feeder vehicle had to follow a non linear trajectory to evaluate the localization reliability also in the presence of turns.
Finally, data are acquired and elaborated with the MatLab and Simulink software to evaluate the estimated pose quality in terms of position and orientation mean, variance and standard deviation error with respect to the true one.
Good results are obtained for the indoor fiducial markers system localization technique with a position and orientation error of 0.3046 [m] and $1.45°$, respectively. On the other hand, for the outdoor RSSI-based Wi-Fi localization approach, estimated pose results in low estimated pose accuracy with a position error of 1.7739 $[m]$.

Keywords: AGV, aluminum smelter, fluoride feeder, indoor localization, fiducial markers system, ARTag, outdoor localization, RSSI, Wi-Fi, wireless sensors networks, extended kalman filter.

Contents

Listing of figures

Listing of tables

1 Introduction

Nowadays, many factories face with labor shortage, high production standard, quality control of products processes, demand for improved productivity and request for qualified workers; consequently, any operation with low human expertise that requires high standardization and precision is complemented support by an automated operation. For example, material handling system is one of the most important area for automation. Moreover, many industrial processes, due to danger factors like high temperatures, gas, dust and work repetitiveness, can be hazardous for human workers. Hence, *Automated Guided Vehicles (AGVs)* are designed to replace conventional systems in order to increase security, efficiency, time management and profits.

Traditionally, AGVs were mostly used in manufacturing systems. Currently, AGVs are also employed for routine transportation tasks in other areas, such as warehouses, container terminals and external (underground) transportation systems [1].

The navigation system is a fundamental component of a modern AGV since it determines its flexibility, reliability and work efficiency. A navigation system has to be able to solve the so called *global localization* problem, meaning it has to obtain the pose (position and orientation) of the related AGV within the work environment. The pose information needs to be retrieved using only data gathered from the surroundings. Information of the surrounding area are obtained using many types of sensors like *laser scanners*, *camera*, *sonars* and *wireless transceivers* which are attached to the AGV itself. Several localization approaches are developed in the recent years, which can be split into three main categories:

1. *Natural Landmarks Localization*, which uses natural feature like building structures (wall, ceiling and columns). This localization technique requires that most of the features does not change over the time[1].

2. *Artificial Landmarks Localization*, which uses reflectors, planar markers and magnets or colored floor stripes[1]. This localization technique requires an invasive setup to be laid down.

3. *Radio Based Localization*, which uses radio signals sent and received by transceiver devices. This localization technique requires a supporting infrastructures.

In the *Indoor Localization*, many natural landmarks can change: for example, the latter can be occluded by other elements, changing the geometry of the environment. Hence, artificial landmarks system is the suitable choice to perform localization on messy environments like factories. Artificial landmarks system is deployed in the environment where the AGV is moving, to accomplish the localization task with high precision, efficiency and reliability. Unfortunately, some of the artificial landmarks localization system due to the high installation and maintenance costs are adopted only by large industries. Moreover, systems like magnetic and color stripes need floor modifications and they stop working when stripes are dirty or damaged. Thus, these restrictions open the way to research and develop the *Vision Based Localization* with *2D fiducial marker systems*, especially if they can be made easier to configure, simultaneously more cost-effective and highly accurate [2].
In the *Outdoor Localization*, natural and artificial landmarks localization systems are, significantly, affected by several environmental parameters which can heavily compromise localization accuracy and events and features. Indeed, changes in lighting conditions during the day can partially or totally obscure landmarks or produce annoying shadows and reflexes, while

badly atmospheric conditions like rain, snow and heavy wind can damage the artificial landmarks already placed in the environment. Moreover, outdoor areas are, usually, very wide and without buildings or structures where, for example, reflectors and markers can be safely and permanently attached. These disadvantageous conditions cause no trivial identification of efficient natural landmarks and high installation and maintenance costs for the artificial ones making these two types of localization not usable. Hence, the *Radio Based Localization* is a valid alternative for the outdoor localization. Indeed, *Global Position System (GPS)* and *Wi-Fi / Radio* are the most used technologies adopted in the last years. The GPS, which is still dominating the outdoor localization market, has been widely used for many applications including autonomous navigation of vehicle and robots. Despite its popularity all over the world, it suffers of high power consumption of its sensors which reduces considerably the battery life of the devices and poor localization accuracy (5-10 [m]). Moreover, many areas on earth are not reachable by the GPS signal or characterized by poor reconstruction features. Thus, Wi-Fi based localization system is often chosen for AGV outdoor localization due to its low costs installation and maintenance and low power consumption. Moreover, as an additional benefit, wireless transmitters placed in the environments (using pre existing structures like lampposts or road signs) can be used to create a communication network between AGVs and their software management exchanging navigation and tasks information.

1.1 Thesis context

In the aluminum smelter plants, the primary aluminum production process takes place. Many tasks are performed, by workers, in hazardous conditions. Thus, many companies involved in the production and distribution of operative vehicles for the aluminum industry, started, thanks also to the technology research and improvements in the autonomous driving sector, to develop AGV vehicles.
One of these company is the *Techmo Car S.P.A.*, specialized in the production of high-end mobile and stationary equipment for the production of aluminium and metal. To achieve this goal, the latter has started a collaboration with the University of Padova, resulting in three thesis that analyze three main AGVs tasks: path planning, obstacle avoidance and localization.

Studies are performed considering a specific aluminum smelter building, the *potroom*, and a specific operative vehicle, the *fluoride feeder*.

1.2 Localization systems

This section gives a short overview of the localization system available on the market. To choose the proper one, studies of installation and maintenance costs, environmental characteristic and dimension of working area needs to be done.

Vision localization system

Using this type of localization system, AGV uses cameras to record natural or artificial features along the path. A vehicle that uses this type of system requires a pre defined map in which the pose of each features needs to be recorded. The extraction of the features from an image (corners, edges, etc.) requires an higher computational power and time compared to all the other localization system but, on the other hand, it not require of any type of wire, stripe and tape to be placed in the environment resulting in cheaper installation and maintenance costs.

Inertial guidance localization system

Inertial guidance system is defined by gyroscopes and accelerometers sensors mounted on the AGV, specifically in the *Inertial Measurement Unit (IMU)* sensor device. They provide, respectively, fixed reference directions or turning rate measurement, angular velocity and linear acceleration changes of the system but suffer from noise and bias that may alter the measurement considerably. Therefore, transponders or corrective markers are placed on the environment to eventually correct the position and orientation of the AGV itself.

Laser scanner localization system

This type of localization is one of the most used in industrial environment and, in particular, for forklift AGVs. A rotating laser scanner (Lidar) mounted on the top of the vehicle determines its position and orientation by precisely measuring angles and distances relative to reflectors placed on wall and columns (triangulation approach). To uniquely retrieve the pose of the vehicle, at least three reflectors have to be detected by the laser to triangulate its exact position. Despite its popularity, it is an expensive localization system because of the high number of reflectors need to be mounted on the environment.

Contour based localization system (SLAM)

Simultaneous localization and mapping (SLAM) is a technology capable of determining the current position of a vehicle without any kind on external infrastructure or artificial landmark placed in the environment. Only safety laser scanner needs to be mounted on the AGV: they are simultaneously used for both localization and safety purposes. The localization is robust against disturbances such as the appearance of other vehicles, people and other dynamically moving objects. On the contrary, once the contour map is created, the fixed objects labeled as landmarks do not have to move or remove from their locations.

Magnetic rods localization system

This approach involves small magnetic rods which are inserted on holes on the floor. Vehicle's route is defined using the floor magnets as key points. Vehicle positioning is calculated based on its prior know position, the distance traveled and direction of travel. Magnetic sensors on the vehicle detect the floor magnets and, on the basis of field strength, calculate the absolute position of the vehicle. Costs of installation and maintenance are proportional to the environment dimension; anyway, this localization system requires floor modification which is forbidden, for example, in the potrooms of the aluminum smelter plants.

Magnetic strips localization system

Magnetic strips, which are attached to the operating floor surface to define the vehicle's operating path, communicate with magnetic-field sensors positioned on the bottom of the vehicle. Magnetic strip not only provides the path for the AGV to follow but also, due to different combinations of polarity, sequence and distance laid alongside the track, tell the AGV to change direction and velocity. Despite its simplicity, this type of localization system need high maintenance costs because of the deterioration of the strips.

Optical localization system

Similar to the magnetic strips localization system, it uses colored strips mounted over the floor to define the vehicle paths. Strips are quite cheaper but they can be damaged fastly in high density traffic areas.

Hybrid localization system

This system merges several localization techniques to obtain better performance with respect to estimate position and orientation. For example, laser scanner navigation can be combined with the Wi-Fi localization to retrieve information about surrounding environment, increasing the AGV adaptability to external events. On the other hand, installation and maintenance costs of hybrid systems are proportional to the number of different localization approaches employed. Moreover, they require sophisticated algorithms to fuse correctly all the needed information acquired from the AGV sensors.

1.3 Proposed solution

This thesis analyzes two solutions for indoor and outdoor localization: the *2D fiducial markers system localization* based on the *AR Tag fiducial markers* and the *RSSI-based Wi-Fi localization* system using the *trilateration* algorithm. Both techniques are tested in a simulated environment, where tags and wireless transceivers are placed, the first in the internal area and the second in the external one. Both localization applications are simulated in stationary and moving conditions, where, in the latter, the AGV needs to follow a nonlinear trajectory to move from an initial point to an ending one. In particular, for the outdoor localization, stationary test is performed using different numbers of wireless transceivers, while the moving one is obtained by choosing two different threshold values for the RSSI.
Indoor and outdoor estimated pose measurements are influenced by uncertainties, which can, significantly, affect the pose of the AGV. Moreover, due to possible temporarily marker occlusion or wireless signal loss, systems was not be able to retrieve the pose of the vehicle for

a certain amount of time. To avoid these serious problems and to increase the estimate pose accuracy, both information obtained with the localization methods and the IMU sensors, on board on the vehicle, are fused together using an *Extended Kalman Filter*.

1.4 Thesis outline

In Chapter 2, a brief description of the aluminum smelter context is given. In particular, the aluminum process and the main buildings of the factory are highlighted. Finally, the Techmo Car reality and the Fluoride Feeder vehicle specifications are depicted.
In Chapter 3, the perception camera model is described, specifying the key matrices involved in the camera calibration and the process to retrieve the relation between 3D coordinates with respect to the world reference frame and the point on the 2D image plane defined in the camera reference frame.
Chapter 4 describes the fiducial markers indoor localization technique and how it's possible to compute the 3D vehicle pose with respect to the 3D world reference frame, starting from the data obtained with a marker detection algorithm.
Chapter 5 shows the RSSI-based Wi-Fi outdoor localization system. In particular, the trilateration process, used to retrieve the vehicle position from RSSI values, is described.
In Chapter 6, the sensor fusion technique based on the Extended Kalman filter is depicted. In particular, a brief description of the IMU sensor and the kinematic model used by the filter to retrieve the vehicle estimated pose are described.
In Chapter 7, the hardware and software used to setup and run the simulation are defined. Moreover, the environment and vehicle modeling process used to create a suitable simulation, are illustrated.
Chapter 8, the last chapter, shown the experimental results obtained in the simulative environment. In particular, stationary and moving simulations are performed for both the localization techniques pointing out mean error, variance and standard deviation of the estimated position and orientation.

2
Context Analysis

In the first part of this chapter a brief description of the aluminum production processes takes place, while on the second part, the Techmo Car company and the operative vehicle considered for this project are depicted. Section 2.1 and Section 2.2 cover, respectively, a general description of the primary and secondary aluminum processes. Section 2.3 and section 2.4 outline the common factory layout characteristics, highlighting the main aluminum smelter buildings and the operative vehicles that work there. Finally, section 2.6 describes the reality of the Techmo Car company.

2.1 Primary aluminum Production

Primary aluminum is produced from *bauxite* ore that is converted into aluminum oxide (alumina). The latter is, finally, reduced to aluminum. The common industrial production practice consists of two consecutive stages:

1. *Bayer* process: in this first step, the high grade metallurgical alumina is produced from bauxite.

2. *Hall-Héroult* process: the second step converts alumina to aluminum by electrolytic reduction.

Both processes were developed at the end of the 19th century and optimized through the years thanks to continue technological improvements [3].

2.1.1 Production of Alumina

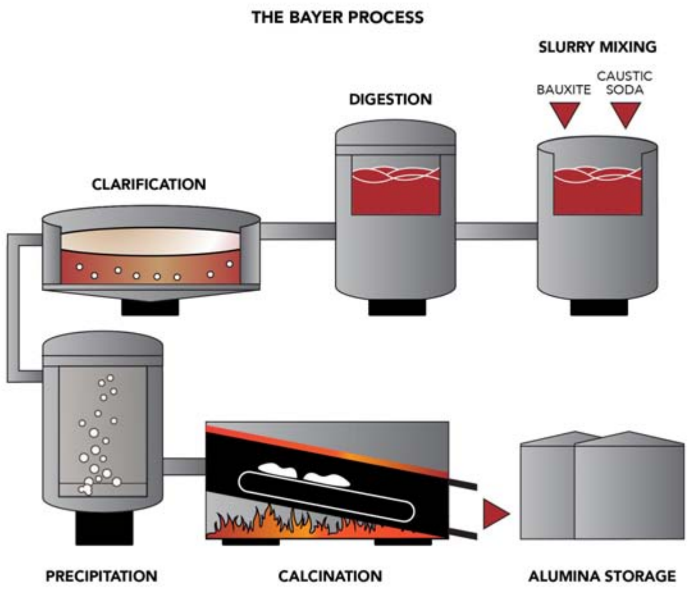


Figure 2.1: Example of Bayer process for alumina refinery

The extracted bauxite ore contains at most 30-60% of alumina, while the remaining 40-70% is composed by unwanted materials such as silicon and titanium dioxide and iron oxide. The latters have to be removed to avoid impurity and metal contamination during the aluminum production. Indeed, with Bayer process, aluminum oxide (Al_2O_3), is extracted from bauxite in a refinery plant. With various modifications, this is the most commonly used method for alumina refining and it involves four steps (Figure 2.1):

1. *Digestion*: in the first step, the bauxite ore is grounded and mixed with a hot solution of lime and caustic soda. The mixture is then pumped into high-pressure containers and heated.

2. *Clarification*: in the second step, separation process dissolves the aluminum oxide by a caustic soda. The process result is a clarified dissolved alumina.

3. *Precipitation*: in the third step, alumina is pumped and added into precipitators where it is cooled down and separated from silicon dioxide. In this step, crystals of aluminium hydroxide are discovered.

4. *Calcination*: in the last step, the agglomerates of aluminum hydroxide crystals are filtered, washed and calcined in rotary kilns at high temperatures. A dry and fine white powder of pure alumina is the result [4].

2.1.2 From alumina to aluminum

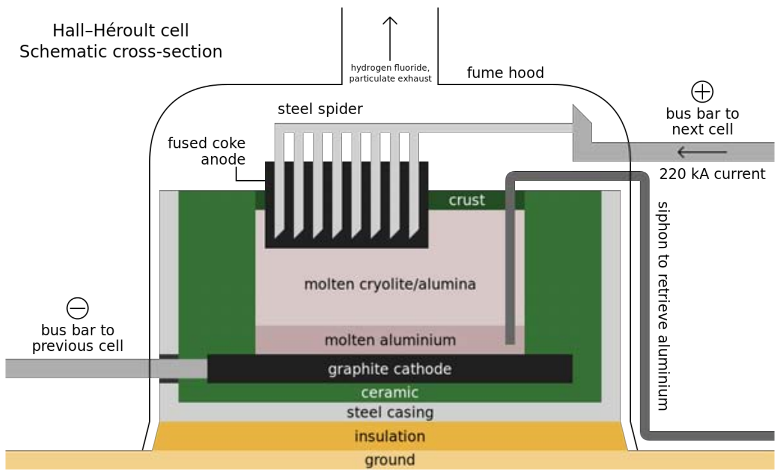


Figure 2.2: Hall-Héroult process for primary aluminum production

Aluminum is produced by the electrolysis of alumina (10 %) dissolved in a molten cryolite-based electrolyte (80 %). The electrolyte is a solution of aluminum oxide in molten cryolite containing an excess of aluminum fluoride (10 %) to decrease the melting point temperature from 2000 °C near to 1000 °C [5]. In the modern aluminum smelter, the molten cryolite-based electrolyte is put into an electrolytic rectangular cell, also know as potcel (Figure 2.2). The temperature to hold the cryolite-based electrolyte bath in a liquid state is obtained by

Joule effect provided by the current passing through the molten bath.. While the voltage between the graphite anode and catode is relative low (4.5 - 5V), the current can easily rise up to 150 - 500 kA causing a chemical reaction inside the potcel. From that reaction, aluminum, carbon dioxide and gases are created: the first one is pumped outside the potcel to cold down and led to other aluminum smelter building to next processing, while the carbon dioxide as particulate exhaust is usually vented to the atmosphere. Finally, the exhausted gases are captured by specialized filters.

2.2 Secondary aluminum Production

Recycled aluminum currently makes up a third of the total aluminum used in the world. Recycling is an essential part of the aluminum industry, given that this process makes economic, technological, and ecological sense (Figure 2.3). The aluminum destined for recycling can be divided in two categories

1. Byproducts obtained during the aluminum process and transformation
2. Scrap of already used old parts that are transformed into ingots and plates for later commercialization

The byproducts have their origin in the manufacturing process of aluminum material (shavings, off-cuts, molded parts, etc.). Usually, their quality and composition is already known. They can, therefore, be melted down without having to carry out any previous treatment and analysis.
The scrap is aluminum material from already produced aluminum goods, which have been used and discarded at the end of their useful life (cables, pots, radiators, etc.). This type of aluminum scrap reaches the recyclers after a considerable number of separation processes. Due to the presence of other undesirable materials, previous treatment and separation is necessary [6]. One of the greatest advantage of the secondary aluminum production is energy saving: the power consumption is 5 % of that one spent in the primary aluminum production process.

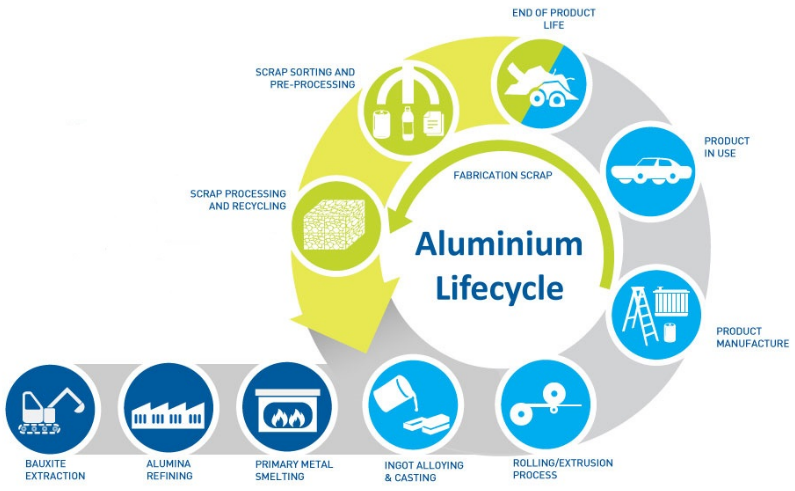


Figure 2.3: Aluminum life-cycle and secondary aluminum production process

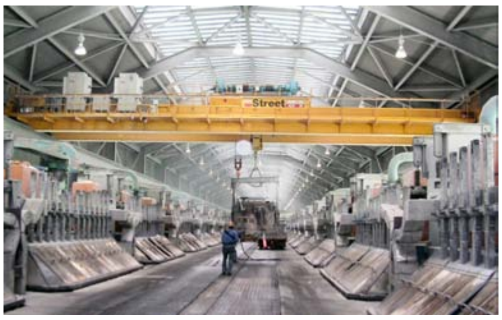
2.3 Factory Layout

The production of primary aluminum takes place in aluminum smelters which consist of large production lines containing hundreds of electrolytic-cells and a number of different common processing areas in aluminum smelters [4]. The most important are:

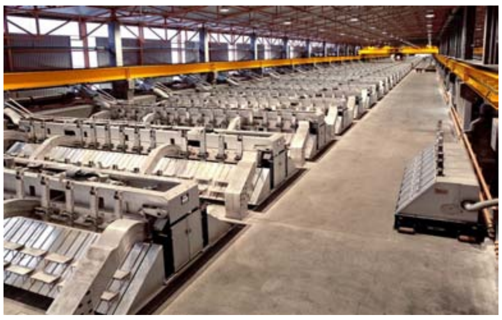
- Potrooms
- CastHouse
- Carbon Anode Baking Shop

2.3.1 Potrooms

The plant layout of a modern aluminum smelter is characterized by a series of parallel long building named potrooms, where hundreds of potcels (100-400 for each potroom) are placed in. Potrooms can be more than 1 km long, in some cases about 50 m wide, and 20 m high. The potcels can be placed “end-to-end” (layout used in the older aluminum smelter), where the short side of two consecutive potcels are faced up, or “side-by-side” where the long side of two consecutive potcels are faced up (Figure 2.4). The latter is the one used in the modern aluminum factories: the advantage in terms of potroom wideness is the presence of only one corridor where the ground vehicles can perform all the needed operations. Usually, just under the potroom ceiling, a crane can be placed in order to move heavy loads or to perform multiple operations on the potcels.



(a) End-to-end layout



(b) Side-to-side layout

Figure 2.4: Potroom layouts

2.3.2 CastHouse

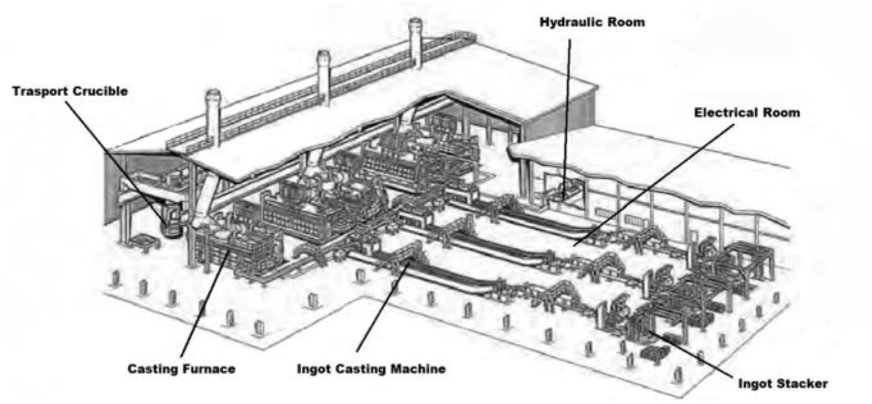


Figure 2.5: Casthouse

The molten aluminum is extracted from the potcels (*tapping* procedure) and transported to the casthouse into crucibles by a crucible transporter. There, the aluminum is inserted in an *holding furnace* to maintain the high aluminum temperature and then place into the *casting furnace* where the primary aluminum is mixed with the secondary one to create ingots (Figure 2.5).

2.3.3 Carbon Anode Plant

Carbon anodes are used in aluminium production in the pre-bake anode process. The carbon anodes are produced in the carbon plant which is a building separated from the potrooms. The carbon anode plant is, usually, divided into three main sections (Figure 2.6):

1. *Green anode*: in this area, the *coke* (calcined petroleum coke) is reduced to a specific particle size and mixed with a solution of liquid petroleum pitch to form a semi-solid compound. The latter is then pressed in order to create a green rectangular anode.

2. *Oven area*: in this section, the green anodes are baked in ovens for several days at a temperature of approximately 1400°C. The volatiles and participates are either kept under pressure in the ovens where they are burned as fuel, or removed and passed through a filter system.

3. *Anode rodding*: in this last area, metallic rods are inserted in the center of the oven carbon anode blocks. The result is a new carbon anode ready to use in the potcels[7].

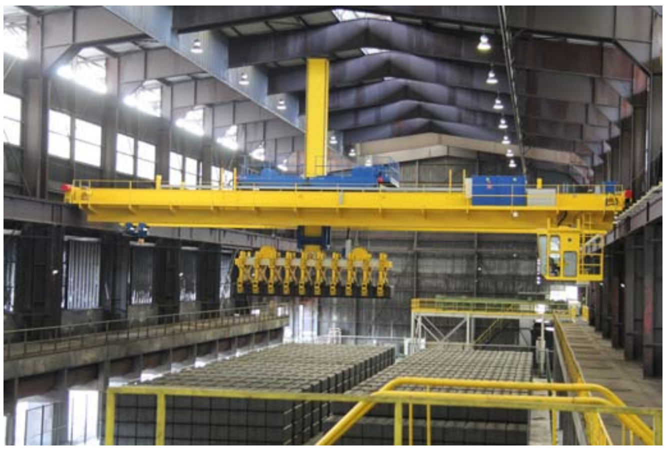


Figure 2.6: Carbon anode plant, anode rodding area

2.4 Operative vehicles

In the older aluminum smelter and, in particular, inside the potroom buildings, almost all the operations made on potcels are performed by cranes. Even if, using only cranes has some advantages in terms of free space increasing and lower number of workers, a crane failure can cause the complete interruption of the related potroom and the interruption of the primary aluminum process, with huge consequences in terms of money loss and wasted material.
In the modern aluminum smelters, most of the crane operations such as tapping, crucible transporting (Section 2.3.2) and potcel fluoride feeding are performed by the so called *operative vehicles*. These vehicles are equipped with specialized arms and components to efficiency replace cranes and to perform operations inside the aluminum smelter. In the next sections examples of operative vehicles are depicted.

2.4.1 Potroom vehicles

End-to-end Potroom vehicles

1. **Anode changers**: they are used to substitute the consumed anodes with the new ones. Furthermore, they adjust the distance between anode and cathode to obtain suitable potential difference and current intensity inside the potcels (Figure 2.7a).

2. **Crustbreakers**: they are used to break the crust present along potcel sides or between the anodes and to simplify the anode extraction and replacement. They are equipped with an hydraulic hammer controlled by the operator (Figure 2.7b).

3. **Alumina feeders**: they are used to fill potcels tanks with alumina (Figure 2.7c). These vehicles have a container of volume between 7 and 9 m^3 which is filled with alumina using a specific charging silo. In the potroom, vehicles are guided to the potcel and fill the alumina tanks using their retractile arm (these operation are almost the same of the fluoride feeder, see Section 2.7).

4. **Anode covering vehicles**: they are used to cover the undercover part of the potcel, after the crust breaker or the anode changing, using a minced melt (recycling material coming from the consumed anodes). This minced melt has a mechanical function:

it thermally insulate the melting aluminum from the external environment and from the oxidation. In some particular cases, a certain amount of alumina is added with the minced melt (Figure 2.7d).

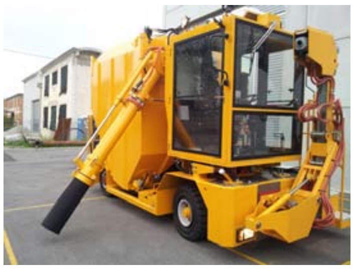
(a) Anode changer vehicle

(b) Crustbreaker vehicle

(c) Alumina feeder vehicle

(d) Anode covering vehicle

(e) FFV


(f) Taphole breaking and covering

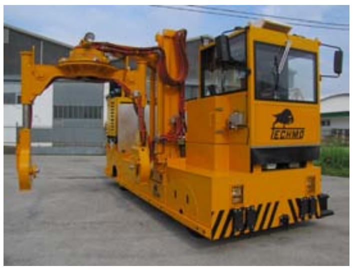

(g) Bath tapping vehicle

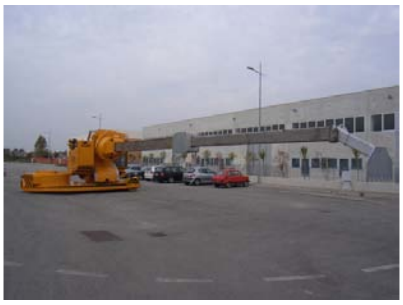
Figure 2.7: Potroom operative vehicles

Side-by-side Potroom vehicles

1. **Fluoride feeders**: similar to the Alumina Feeder (Section 2.4.1) they perform feeding operations which are deeply describes in Section 2.7 (Figure 2.7e).
2. **Taphole breaking and covering**: they used an hydraulic hammer to open hole on the crust of the short side of the potcel which permits the molten aluminum extraction. Then, the hole is closed using a minced melt (Figure 2.7f).
3. **Bath tapping vehicles**: they are used to balance the cryolite bath quantity in order to create uniform aluminum production conditions. This process consists of continuously extracting and adding the cryolite bath from from specific potcels to other ones (Figure 2.7g).

2.4.2 Casthouse vehicles

1. **Skimming vehicles**: they are equipped with a telescopic arm on which a metal rack is mounted on the extremity. They are used to remove the superficial metal waste, to mix the molten metal and to clean the bottom and the walls of the furnace.
2. **Scraps loaders vehicles**: they are used to charge the furnaces with aluminum scrap. To accomplish this work, the scraps loader vehicles use forks and other loading devices.


(a) Skimming vehicle

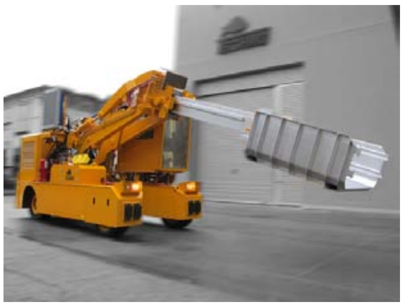
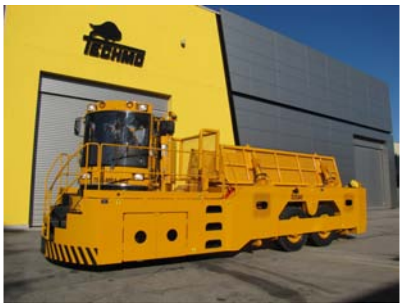

(b) Scraps loaders vehicle

Figure 2.8: Casthouse operative vehicles

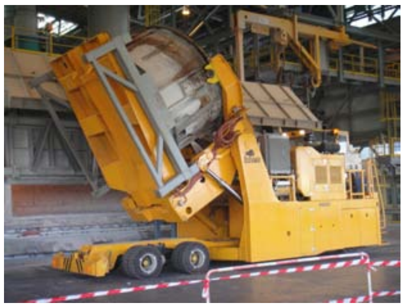
2.4.3 Other Vehicles

1. **Anode transport vehicle** (ATV): this vehicle performs tasks in different buildings of the aluminum smelter. It transports anodes pallet, which contain either both new and old anodes, from the carbon anode plant (Section 2.3.3) and the rodding shop to the potroom. Pallets are lifted up by the ATV specific hydraulic system after a precisely backward maneuver (Figure 2.9a).

2. **Crucible transport and tilting vehicle** (CTTV): it transports crucibles from the potroom to the casthouse (Section 2.3.2). The molten aluminum contained in each crucible is inserted in the holding furnace by a pouring or siphoning maneuver provided by the CTTV (Figure 2.9b).



(a) Anode transport vehicle



(b) Crucible transport and tilting vehicle

Figure 2.9: Anode transport vehicle and Crucible transport and tilted vehicle

2.5 Hazards and Risks of the Primary aluminum Process

Potrooms and, in general, the aluminum smelter buildings, are dangerous work places for operative vehicles and workers. Many hazardous tasks are performed each day in critical conditions, which can cause several physical and mental injuries to the workers. Moreover, operative vehicles check up needs to be done periodically to prevent damages caused by the usury of the mechanical and hydraulic components in such a critical environment. Indeed, the smelter environment is characterized by severe conditions in terms of:

Temperature: regardless the place where the primary aluminum smelter is building, inside the potrooms temperatures can easily achieve 50-60 °C and, often, in proximity of the potcels, can rise up to 80-90 °C. Hence, vehicles cabin are equipped with air conditioning and all the components that can be damaged by high temperatures are covered with special protections or cooled down by specific cooling systems.

Magnetic Field: due to the high intensity of the current used for the aluminum process, inside the potrooms magnetic field can achieve values of 0.3 - 0.4 Tesla. This can cause serious problems to the ferromagnetic mechanisms. To prevent dramatic consequences some precautions can be take in place. For example, sensible parts of the vehicles should be put on the top of them and the vehicle frame is used as a protection. Furthermore, the potroom access is forbidden to all the people with pacemaker.

Electric Energy: due to the current intensity, potroom machines have to work in an electric environment. If a vehicle accidentally touch a potcel, due to its high mass and its metal structure, transforms itself into a huge resistance subject to a Joule effect (the vehicle can be burned causing serious hazard to the operator). To avoid it, all the machines are electrically isolated (large plates of insulating material are mounted between frame and moving parts).

Dust: alumina and fluoride powders are very dangerous, in particular for drivers health.

Because of that, machines are equipped with specific filters and seals to maintain engine and cabin dust-free.

Gas: fluorine gas can ruin glass. Thus, vehicle glasses are protected with particular film or are made in poly carbonate. Furthermore, some aluminum companies ask for filter and ventilation systems to protect the driver, removing all the nocive gas inside the cabin.

Drivers: drivers are, probably, the biggest hazard for themselves and the plant. They work as fast as possible ignoring the vehicle status. Moreover, to increase work efficiency, they bypass most of the vehicle security systems, which are necessary for ensure driver safety when something goes wrong. Finally, driver who perform repetitive tasks can soon become distracted and and not careful.

2.6 Techmo Car S.p.a.

Techmo is a world leader company in the engineering and production of high-end customized mobile and stationary equipment for the primary aluminium industry. The company was founded in 1961 by Dr. Franco Zannini and since the beginning it was focused on providing original and state of the art solutions to problems related to metal production. Techmo was the first company in the world to create and fabricate specialized vehicles for the aluminium electrolysis in the 1960's. Nowadays, Techmo products, located in more than 40 countries, are appreciated by the most demanding aluminium producers, adopting every kind of smelter reduction technologies. Techmo research and development programs focus on the following multiple objectives:

Figure 2.10: Techmo Car S.p.a. logo

- Improving all the environmental aspects: better comfort and safer working conditions for the operators, reduction of carbon footprint of the equipment and increase of the recyclability of their components

- Bringing Higher efficiency to the production process while reducing production costs, running costs and maintenance costs

- Increasing durability in the years

To achieve all of these objectives, Techmo is now studying the possibility to adopt the AGV technology and implement it to create safer vehicles reducing production, running and maintenance costs all over the years. Techmo has already started some research to apply this technology in the aluminum industry: a collaboration with University of Padova is focusing on applying this new technology to a specific vehicle, the *Fluoride Feeder Vehicle* (Section 2.7).

2.7 Fluoride Feeder Vehicle

One of the most iconic operative vehicle of Techmo Car S.p.a. is the **Fluoride Feeder vehicle** (FFV). It is a six-wheel equipped diesel truck with retractile arm (Figure 2.11) which performs its operations in the potrooms and in the Aluminium Fluoride Storage and Handling (AFSH) areas (see Table2.1 for general specifications). It specifically performs two tasks:

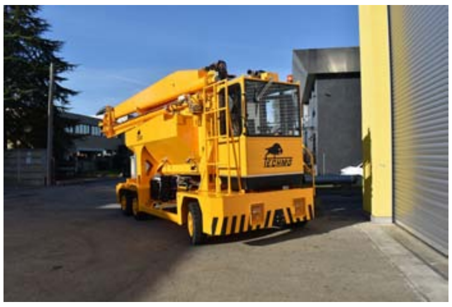
- Feeds the potcels with aluminum fluoride
- Charges its container in the AFSH

2.7.1 Potcels feeding

Its main task is to feed each potcel with the aluminum fluoride (Section 2.1). When a potcel required a certain amount of aluminum fluoride to accomplish the primary aluminum production, the FFV's worker drive until he reaches the selected potcel.
Then the FFV retractile arm, initially positioned over its container, starts moving outside the vehicle until it reaches the desired position over the potcel. Finally, thanks to worm inside the container and the arm, the aluminum fluoride goes into the potcel passing through two top openings. Note that, due to electric and hydraulic pipes which run over the potcel, the FFV perform the feeding operation positioned between two consecutive potcels (Figure 2.12).

(a) Left side view of FFV

(b) Right side view of FFV


(c) Front side view of FFV

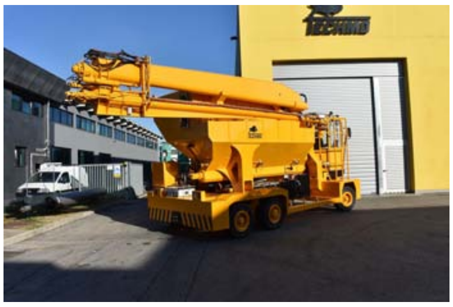
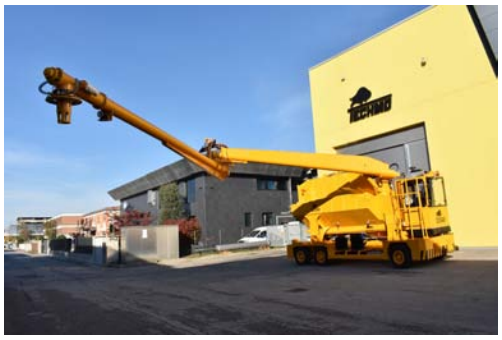
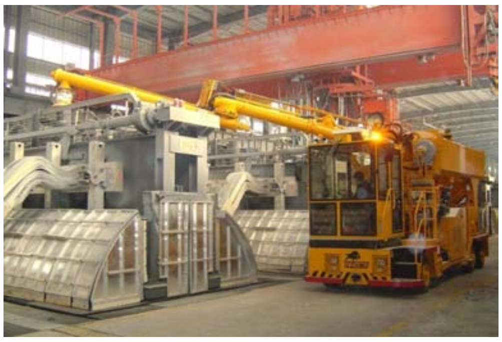
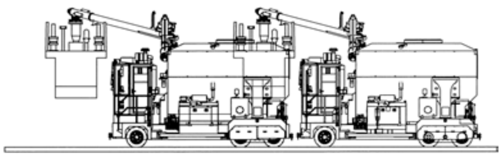

(d) Back side view of FFV

Figure 2.11: FFV


(a) Retractile arm during feed operation


(b) Potcel feeding operation


(c) Lateral view of feeding operation

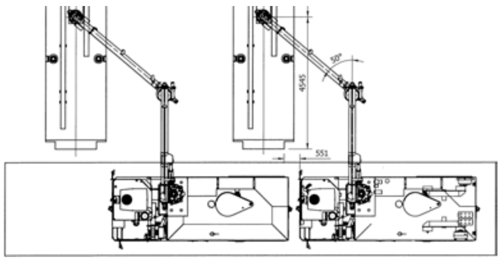

(d) Potcel feeding operation

Figure 2.12: Top view of feeding operation

2.7.2 Container charging

When the container is almost empty, the FFV goes into the AFSH building to charge. The charging station consists of a silos which has an opening on the bottom from which the aluminum fluoride passing into the vehicle. Because of the presence of many FFV (the number depends on the size of the aluminum smelter), this charging operation needs to be perfectly managed to guarantee no waste of time.

FFV Main Specifications	
Length	6100 [mm]
Width	2200 [mm]
Height	3638 [mm]
Tare weight	12500 [kg]
Gross weight	23000 [kg]
Container capacity	10500 [kg]
N° of wheels	6
N° of traction wheels	2 wheel drive on the rear axis
Type of engine	Diesel engine
Minimum steering radius	6150 [mm]
Wheel diameter	720 [mm]
Wheelbase	4180 [mm]
Retractile arm's maximum yaw angle	120 [deg]
Retractile arm's maximum pitch angle	37.4 [deg]

Table 2.1: FFV main specifications

3 Perception

As introduced in Chapter 1, the indoor localization is based on fiducial markers (Chapter 4) which are seen by monocular cameras placed over the FFV. To fully understand how this localization procedure works, this chapter describes the frontal pinhole camera model and how extrinsic and intrinsic camera parameters are retrieved from camera calibration.

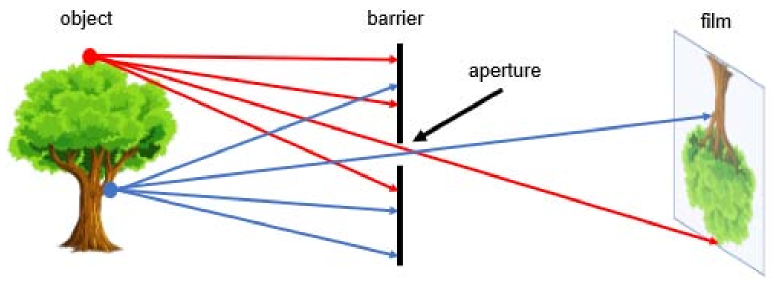


Figure 3.1: Pinhole camera

3.1 Pinhole Camera

The pinhole camera model describes the mathematical relation between the coordinates of a point in three-dimensional space and its projection onto the image plane of an ideal pinhole camera (Figure 3.1). This model is only a simple approximation of a modern camera: indeed lens effects and other non-idealities are not taken into account.

3.1.1 Pinhole Camera Model

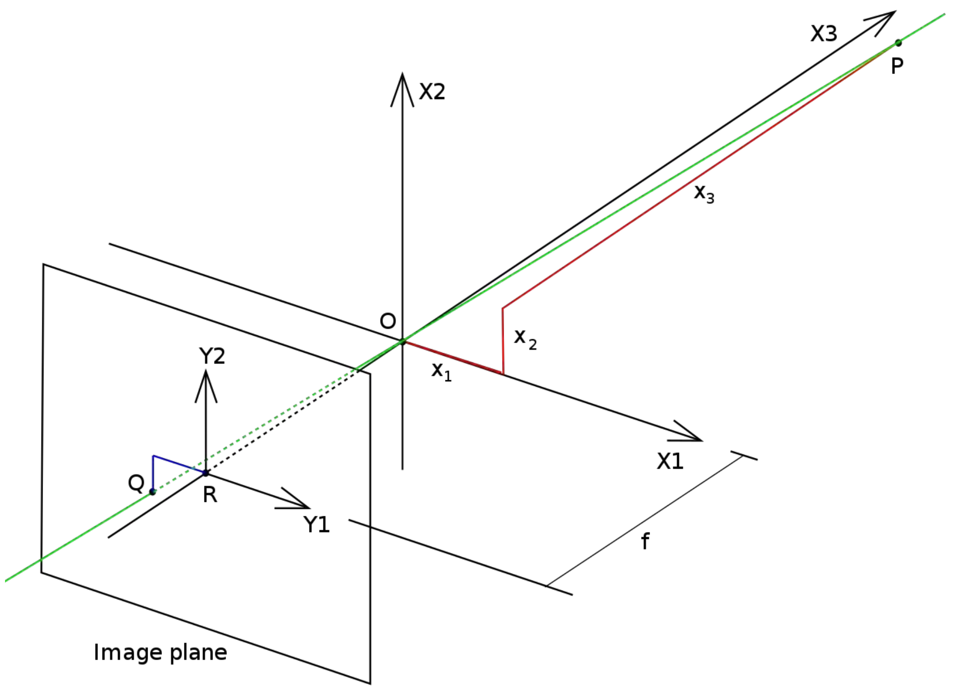


Figure 3.2: Pinhole camera model

Drawing inspiration from Figure 3.2, a generic pinhole camera model is characterized by the following elements [8]:

- *3D orthogonal frame* defined by the triplet (X_1,X_2,X_3) where O is the origin of the coordinate system
- A point P with coordinates (x_1,x_2,x_3) related to the 3D frame

- *2D orthogonal frame* defined by the pair (Y_1,Y_2) which represent the projection of P onto a plane called *image plane*.
- R, at the intersection between optical axis (orthogonal segment from O to the image plane) and the image plane
- *Projection line* of point P into the camera
- *Q* with coordinates (y_1,y_2), which is the projection of P into the image plane
- *f*, which is the distance between O and R, called *focal lenght*

To understand the transformation from the 3D to 2D coordinate system, let's consider Figure 3.3 where P is the point in the 3D world frame and Q is its projection on the image plane. According to the similarity of triangles criteria the relation between P with coordinates (x_1,x_2,x_3) and Q with coordinates (y_1,y_2) is obtained by the following formula

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.1}$$

Note that the image on the image plane of a real pinhole camera is rotated by π. To produce an unrotated image the image plane can be intersect the X_3 axis instead of at $-f$ at f (*frontal pinhole camera model*). The result mapping is

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.2}$$

which leads to

$$\begin{cases} y_1 & = f\dfrac{x_1}{x_3} \implies \dfrac{x_1}{x_3} = \dfrac{y_1}{f} \\ y_2 & = f\dfrac{x_2}{x_3} \implies \dfrac{x_2}{x_3} = \dfrac{y_2}{f} \end{cases} \tag{3.3}$$

Drawing inspiration from Equation 3.3, starting from the 2D coordinates y_1 and y_2, it's possible to retrieve only the 3D coordinate ratio $\frac{x_1}{x_3}$ and $\frac{x_2}{x_3}$. So, using a single camera, it's impossible to perform the inverse transformation from 2D image plane to 3D world frame coordinates without information loss: for example, it is impossible to find the distance from

P to the image plane.

In general, in order to obtain the position of a given point in the 3D frame starting from 2D frame, it is necessary to use *stereo vision system* and *epipolar geometry*. However, in particular cases like markers detection where dimension, color and shape of tags are a priori well known, a single camera model can be used to retrieve the position and orientation of the tag if at least three correspondences between image points and objects points have been identified [9].

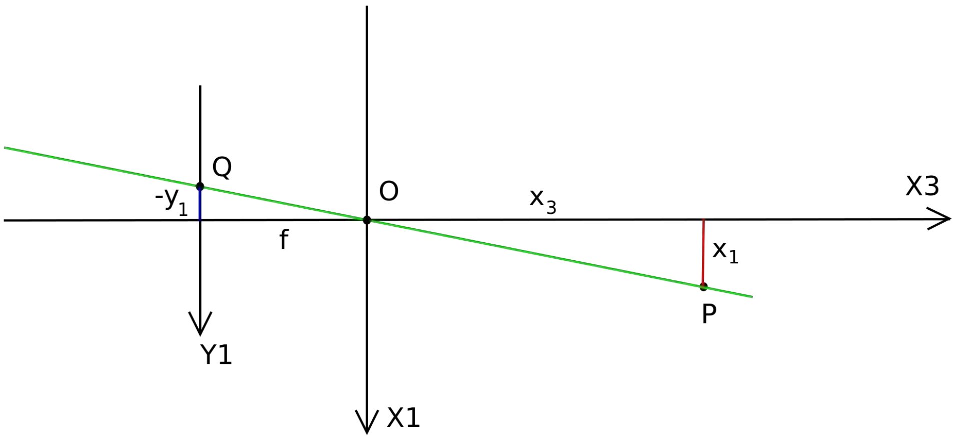


Figure 3.3: Relation between 3D world frame and 2D image plane coordinates

3.2 Camera Calibration

Geometric camera calibration is the process of estimating *intrinsic* and *extrinsic* parameters of a given camera. Intrinsic parameters are, for example, focal length, skew, lens distortion and image center while, the extrinsic parameters describe position and orientation of the camera in the 3D world frame. Accurate camera calibration is necessary to many vision-based 3D metrological techniques. Usually, camera calibration methods using regular planar calibration targets, such as, check-board or circular patterns (Figure 3.4) of

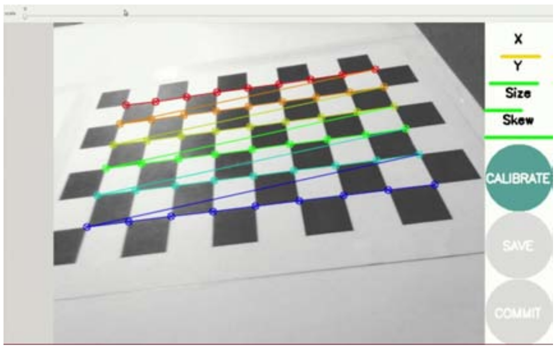


Figure 3.4: Camera calibration method

which dimensions are a priori known[10]. Intrinsic parameters matrix, $\mathbf{K} \in \mathbb{R}^{3\times3}$, is defined as follows

$$\mathbf{K} = \mathbf{K_s K_f} = \begin{bmatrix} S_x & S_y & O_x \\ 0 & S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fS_x & fS_y & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where

- S_x and S_y are pixel scale factors
- S_θ is the skew factor which takes into account the fact that the pixels may not be rectangular
- f is the focal length
- (O_x, O_y) is an offset vector

Let us define the *standard perspective matrix* $\Pi_0 \in \mathbb{R}^{3\times4}$

$$\mathbf{\Pi_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

which maps the homogeneous coordinates from 3D world to 2D world. The relation between the 2D reference frame and the 3D one can be defined using the following equation:

$$\begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{K\Pi_0} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \tag{3.6}$$

where the vector $\begin{bmatrix} x_1 & x_2 & x_3 & 1 \end{bmatrix}^T$ represents the 3D homogeneous coordinates defined on the 3D camera reference frame. Finally, the relation between the point in the 3D coordinates with respect to the world reference frame and the point on the 2D image plane defined in the camera reference frame can be derived considering equation 3.6 and the extrinsic parameters matrix $\mathbf{G}$, that is

$$\begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{K\Pi_0} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} = \mathbf{K\Pi_0 G} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}, \quad \mathbf{G} \in SE(3) \tag{3.7}$$

where $\mathbf{G}$ and $\mathbf{P}$ are the extrinsic parameters matrix and the camera matrix respectively: $\mathbf{G}$ is composed by the 3×3 rotation matrix $\mathbf{R}$ and the 3×1 translation vector $\mathbf{T}$, which map the vector $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ defined in the 3D camera coordinates, into the vector $\begin{bmatrix} X_1 & X_2 & X_3 \end{bmatrix}^T$ defined on the 3D world reference frame according to:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} = \mathbf{G} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \mathbf{T} \tag{3.8}$$

4 Fiducial Markers Indoor Localization

For industrial purposes, many vision applications use two dimensional patterns to carry information used to perform accurate AGVs localization and navigation. *Fiducial markers* system consists of artificial planar patterns that are mounted in the environment on pre existent structures and automatically detected by cameras using suitable algorithms. They are used for several scientific and technical purposes and, in particular, robotics, augmented reality, navigation, localization and camera calibration. They combine fast and accurate camera position and orientation estimation with easy and inexpensive installation and maintenance[11]. The robustness and usefulness of a fiducial markers system are characterized by several qualitative and metrical parameters, that are:

1. *False positive rate*: rate of the erroneous reporting the presence of a tags when none is present

2. *False negative rate*: probability that a marker is present in an image but not reported

3. *Inter-marker confusion rate*: rate of wrong id extrapolated from the detected markers

4. *Vertex jitter characteristics*: noise in the marker corner positions, which affects the stability and accuracy of the tags detection process

5. *Marker library size*: high number of unique markers that the system can support

6. *Minimal marker detection size*: is the size in pixels required for reliable detection

7. *Immunity to lighting variation*: markers need to be detected correctly even in low light environment

8. *Immunity to occlusion*: markers need to be detected correctly even if its pattern is partially covered

9. *Speed performance*: a vision-based fiducial marker localization system needs to achieve real time performances with low cost computing power [12].

Nowadays, thanks to the technology improvement and research, especially on the augmented reality, multiple planar marker systems have been created, each one characterized by different pattern design: Data Matrix, Maxicode, QR, ARStudio, ARToolkit and ARTag are some examples. Therefore, proper fiducial marker system comparison is necessary in order to choose the suitable one for the task of interest (Figure 4.1)[13].

For its versatility, performances and simple implementation on the simulation setup of this project, the *AR Tag* fiducial markers system has been chosen for the indoor localization.

4.1 ARTag fiducial markers system

ARTag is a planar pattern marker system that counts 2002 markers on its library. Speaking of robustness and usefulness, the false positive rate is estimated to be $< 0.0039\%$ of the quadrilaterals found in an image. The internal marker confusion rate is minimal and vertex jitter has a standard deviation of 0.03 and a maximum of 0.09 [*pixels*]. Moreover, it can be detected despite lighting changes and markers can be detected if partially occluded by external environmental elements. Finally, ARTag markers have square shape providing four

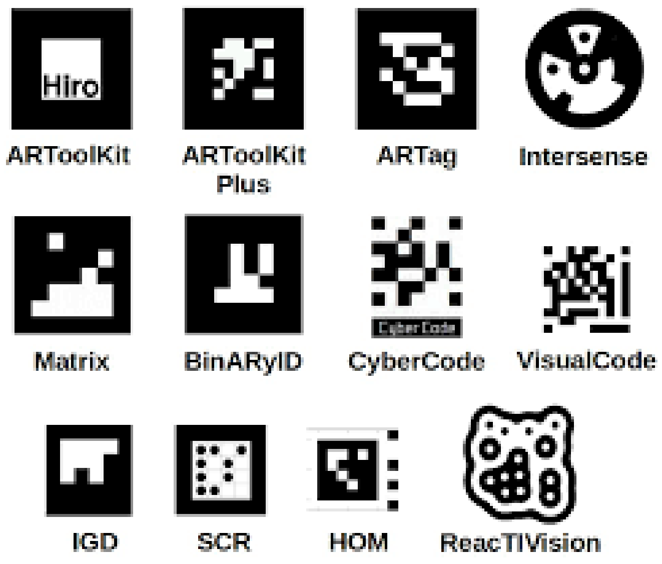


Figure 4.1: Example of planar pattern marker systems

corners necessary for perspective support and tags detection speed is typically on the order of $10 - 50$ [ns], fast enough for real-time performance [12].
This type of marker uses a concept of squares with an internal image, where the latter is read and decoded using a digital approach. The main characteristics of these tags are a square border of either polarity (white on black or black on white) and a 5×5 [u] square grid dividing up the interior. The whole marker is 9×9 [u], with a border of thickness 2 [u] leaving the 25 internal cells to carry information (Figure 4.2). Each cell is only black or white and carries one bit of digital data [14]. The sequence of 25 bits obtained by the internal cells is composed by the *id-bits*, of length 10 bits, used to retrieve the correct ID of the related tag and the *redundant-bits*, of length 15 bits, : the first one, of length 10 bits, is used to detect and correct errors and to insure the uniqueness of the tag ID[15]. Once the markers are placed in the environment in their pre-defined position and orientation, to obtain suitable indoor localization, two processes need to be performed or by cameras with their embedded acquisition system or by vehicle localization and navigation system:

- Markers detection
- Identification and ID decoding

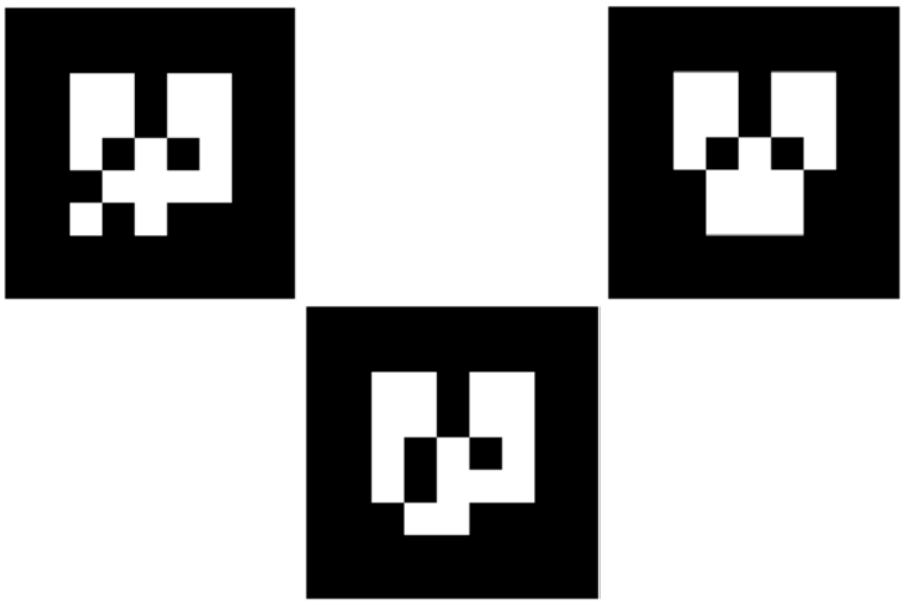


Figure 4.2: ARTag fiducial markers system

4.1.1 Markers detection

The first goal of the marker detection process is to find the outlines of potential markers and then to deduce locations of marker's corner in the camera frame. Moreover, detection system needs to confirm correct marker and retrieve its identity or discard false ones. Finally, the system computes the tag position and orientation with respect to the camera frame using information from the detected marker location (dimensions, pose with respect to the world reference frame, etc.) and from the camera calibration (intrinsic and extrinsic matrices) (Section 3.2). The markers detection process involves several phases which are performed by using standard algorithms or particular libraries (OpenCV):

- Image acquisition
- Pre processing
  - Low level image processing
  - Line detection and line fitting
  - Detection of the marker corners and its quadrilateral

- Markers detection with acceptance criteria
  - Fast rejection of false markers
  - Fast acceptance of true markers

- Identification and ID markers decoding
- Markers pose with respect to the camera reference frame

As already mentioned in the Chapter 1, visual localization approach can be used efficiently inside the aluminum smelter due to its reliability and immunity against several environmental hazardous parameters like high temperatures, corrosive gases, magnetic fields and dusts. Moreover, markers can be placed on the walls and, thus, not damaged by the operative vehicles.

Image acquisition

It's the first process in marker detection. Specific cameras mounted on the FFV acquire frames and send it to the processing and detection system. To increase localization accuracy performances, cameras needs to be mounted in crucial pre defined positions (on the top of the vehicle cabin) in order to increase cameras field of view and to avoid occlusion by the vehicle itself. Moreover, high camera resolution and frame rate values are important parameters for low noise images: the latters need to be set properly without losing real time performances overloading the localization system.

Pre processing

Markers detection is computed on gray scale image that is converted starting from the one obtained by the cameras. A gray scale image is one in which the value of each pixel representing only an amount of light, that is, it carries only intensity information. Gray scale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray from black (low light intensity) to white (high light intensity). If camera frame has different color space format (*RGB*, *CMYK*, *HSV*, etc.), the image is converted into the intensity image. The first operation of the marker detection is to retrieve a *binary image* (pixels have only black or white color). A binary image is also called *bi-tonal* or *two-level* image obtaining by the so called *image segmentation process* which is used to partitioning an image into meaningful regions (in this case the number of region is equal to 2). To correctly separate the two regions, an adaptive threshold technique is applied to compute the best threshold value in a way that minimizes the variance of values within each region and maximizes the variance of values between the regions (*Otsu's algorithm*). The latter guarantees marker detection even with different local illumination changes[16].

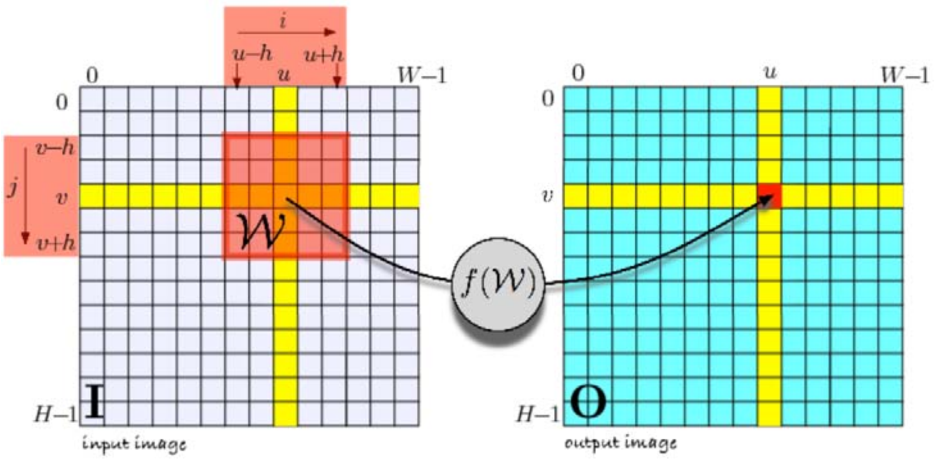


Figure 4.3: Spatial operation

To detect edges into a given image, the gradient direction and magnitude of each pixel are computed using *spatial operation*. With spatial operation, each pixel in the output image is a function of all pixels in a region surrounding the corresponding pixel in the input image (Figure 4.3), that is

$$O[u,v] = \sum_{(i,j)\in W} I[u+i, v+j]K[i,j], \quad \forall (u,v) \in I \tag{4.1}$$

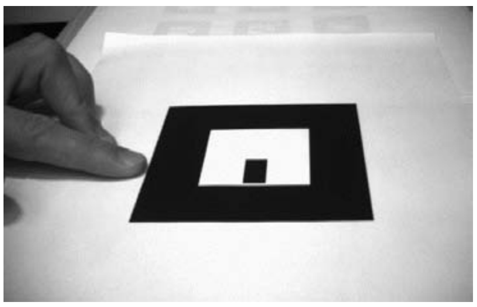
where W is know as window, typically a w $\times$ w square region and $K \in \mathbb{R}^{\mathrm{w}\times\mathrm{w}}$ is the so called *convolution kernel*. For every output pixel, the corresponding window of pixels from the input image W is multiplied element-wise with the kernel K. Many convolution kernels have been proposed and the most common is the *Sobel Kernel*, defined as follows:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Pixels with the same gradient direction and magnitude values are clustered into *components*. Components are generated following a *graph-based recursive segmentation method*: a graph is created where the nodes represents pixels. Adjacent pixels are connected by edge and its weight is equal to the pixel's difference in gradient direction. If the edge weight is under a certain threshold, the terminal pixel is added to the same component of the initial ones (Figure 4.4). Note that, taking the derivative of a signal increases noises, especially to the high-frequencies. To reduce these noises, a *smoothing* operation is applied, by a convolutional operation between the image before the spatial operation and the following *Gaussian kernel* [17]

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\dfrac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

Edges detection is the most time expensive process of the markers detection. To increase


(a) Camera frame

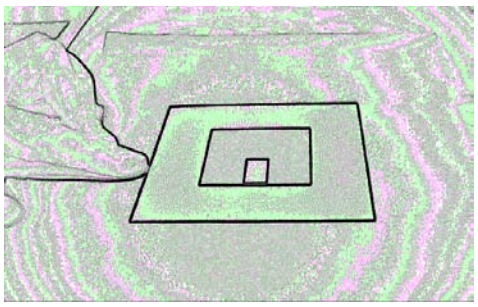

(b) Edge detection

Figure 4.4: Example of edge detection process

speed performance it is possible to reduce camera resolution but, as consequence, small tags could not be detected [18].

At this point, a set of segments have been computed from an image. The next step is to find a close-loop with four corners in order to obtain a 4-sided shape (*quadrilateral*), using a recursive algorithm. The first phase of quadrilateral detection process starts considering all the segments founded in the edges detection. Then, on the second phase, only the segments that begin close enough to the previous segments end and that obey to a counter clockwise winding order are selected. Once the 4 lines are founded, the candidate quadrilateral is created. The next step, before the marker decoding process, is used as first markers selection in order to send to the marker detection step true and valid tags as much as possible (Figure 4.5)[18][19].

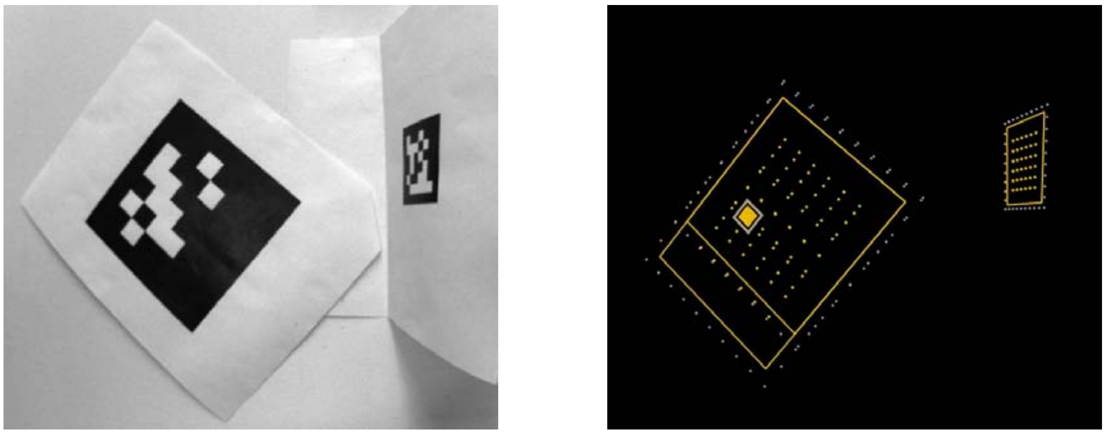

(a) Camera frame **(b)** Quadrilateral detection

Figure 4.5: Example of quadrilateral detection process

Fast acceptance and rejection criteria

Even small error in detected 2D locations of edges and corners might significantly affect the calculate pose of the marker. Detection errors can be caused by pixel quantization error, wrong threshold value, camera noise, etc. All of these errors cause annoying oscillations in the tag's pose. To increase accuracy, detection systems optimise the locations after initial detection.

For example, if marker has been detected using binary image, on the second step, system can used gray-scale image to find edges and corners. Moreover, it can use the first detected corners as initial estimate for a more accurate corner detection method.
Due to high speed performance required in real time augmented reality, markers needs to be accepted or rejected quickly. Thus, fast acceptance/rejection criteria to distinguish real markers from common objects are applied. First of all, system can reject quadrilaterals with an area of a few pixels. Indeed, small area size means that markers is far away from camera or the related quadrilateral is not a marker. To compute size of a certain area, it is possible to calculate the number of pixels which belong to the perimeter of the related area. Another criteria is based on the overall appearance of the markers. Indeed, if the system already known the shape and pattern of the tags, it can reject quadrilaterals which have, for example, all pixels black. Finally, 2D binary markers have a number of strong edges inside the marker (edges between white and black cells). One method of rejecting obvious non-markers is to calculate the number of intensity changes in two perpendicular directions. If the number of changes is low, it cannot be a marker.

4.1.2 Identification and ID decoding

Once the quadrilateral border contours of the markers have been detected, a 5x5 unit spatial windows is used to sample the internal region assigning values 0 or 1 if the color inside each cell of grid is black or white respectively. Performing this sampling for all the possible marker rotations, four 25-bits sequences are obtained; only one of them may end up being validated in the decoding process. The 25-bit binary sequence encoded in the marker, as already mentioned at the beginning of the ARTag section, encapsulates a 10-bit ID which is used to identify the related marker. The extra 15 bits provide redundancy to reduce the false detection rate and to provide uniqueness over the four possible rotations. If any of the four 25-bits sequences is found in the ARTag system, the candidate marker is consider as a valid tag; to speed up this process, the dictionary elements are sorted as a balanced binary tree. To this aim, markers are represented and sorted by the integer value obtained by concatenating all its bits. Thus, the computational complexity of this process is $O(4log_2(|D|)$, where D is the ARTag dictionary. Conversely, if no match is found, a *Forward Error Correction (FEC)* and a *Cyclical Redundancy Check (CRC)* digital methods are applied to the four 25-bits sequences to identify if the related code is part of the ARTag marker set and to extract its ID [20].

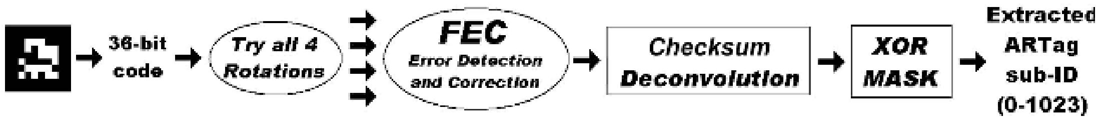


Figure 4.6: Example of decoding process

FEC is used to increase the positive rate repairing some bits of the sequence using the *Hamming Algorithm*. Hamming algorithm is a method widely used for error detection and correction. It is based on the use of *parity bits*. A parity bit tells whether the number of ones in a binary number is odd or even. *Even parity* is equal one if the number on ones is odd and zero if the number of ones is even. Vice versa, *odd parity* is respectively one if the number of ones is even and zero if the number of ones is odd. With only one parity bit added to the data, it is possible to detect one bit error without information about which bit is wrong. On the contrary, with more parity bits, the Hamming algorithm is able to detect the locations of detected errors and, thus, correct them. Note that, using only the FEC process with Hamming algorithm, the system is able to detect and correct only a certain amount of errors within a cells block. In some cases, for example reflections and shadows, the probability error existence between neighbouring cells is correlated. To increase FEC performance uncorrelating the detection error between near cells, *data randomizing* process is applied: the system scatters the bits of each block within the marker. This means that data cells belonging to a block are at different parts of the markers and not adjacent, making the probability error of each marker cell independent to the other one. The latter technique, applied with the data repetition, improves the probability of getting correct bits sequences [16].

Finally, the CRC digital method is applied in series to the FEC to extract the correct marker ID, if the latter is a valid one. CRC is a checksum algorithm to detect inconsistency

of data, i.e. bit errors over a bits sequence. This method is based on polynomial division. The binary input sequence is interpreted as *binary polynomial* (bits are used as coefficients) which is divided by another fixed binary number called *generator polynomial*. The latter is statically defined by the used CRC algorithm: CRC-8 using a fixed defined generator polynomial with $n+1$ bits, that is, for example, $x^8+x^5+x^2+1$. The remainder of this division is the *checksum value*. Division of polynomials differs from integer division. Indeed the CRC process arithmetic calculation is based on the *XOR* (Exclusive-OR) operator, which truth table is summarized in Table 4.1.
CRC method used for the ARTag markers detection is called *CRC-16* and it has as generator polynomial $x^{16}+x^{12}+x^5+1$, which corresponds to the bits-sequence 10001000000100001. To perform the division, 16 bits are appended to the dividend and the most significant bit of the divisor is aligned with the dividend one. If the reminder of this division is 0, the tag related to the used 25-bit sequence is labeled as valid and its ID is extracted and combined with its border polarity. Otherwise, the tag is labeled as not valid and discharged by the system [21].

XOR	**0**	**1**
0	0	1
1	1	0

Table 4.1: XOR truth table

Markers pose with respect to the camera reference frame

To perform fiducial markers localization, once the tags are labeled as valid and their ID have been extrapolated using the above process, the position and orientation of the markers with respect to the 3D camera reference frame are computed. The four corners of the valid tag, with 2D camera coordinates x_c and y_c, are transformed, using the intrinsic camera parameters matrix, into the X_c, Y_c and Z_C coordinates on the 3D camera reference frame. The entire process is analyzed in Section 3.2.

4.1.3 FFV pose with respect to the world reference frame

The goal of both the indoor and outdoor localization approaches is to retrieve position and orientation of the FFV with respect to the world reference frame, meaning finding the rotation matrix and translation vector $\mathbf{R}_v^w$ and $\mathbf{T}_v^w$, respectively. In this section, the technique adopted to retrieve the 3D position and orientation vehicle coordinates with respect to the world reference frame is depicted. To simplify the description of the entire process, let us divide it in three fundamental steps:

1. Compute the vehicle pose with respect to the camera reference frame
2. Compute the vehicle pose with respect to the tag reference frame
3. Compute the vehicle pose with respect to the world reference frame

Notation

Firstly, let us consider N different tags and only one of the cameras placed over the cabin of the vehicle (the process is the same for the other two cameras). Define the following translation and rotation matrices:

- $\mathbf{R}_c^v$ and $\mathbf{T}_c^v$ are, respectively, the rotation and translation matrices from the camera reference frame to the vehicle reference frame
- $\mathbf{R}_c^{t_i}$ and $\mathbf{T}_c^{t_i}$ where $i = 0, ..., N-1$ are, respectively, the rotation and translation matrices from the camera reference frame to the i-th marker reference frame
- $\mathbf{R}_{t_i}^w$ and $\mathbf{T}_{t_i}^w$ where $i = 0, ..., N-1$ are, respectively, the rotation and translation matrices from the i-th marker reference frame to the world reference frame
- $\mathbf{O}_w$, $\mathbf{O}_t$, $\mathbf{O}_c$ and $\mathbf{O}_v$ are, respectively, the origin of the world, tag, camera and vehicle reference frames

Finally, note that $\mathbf{R}_i^j \in \mathbb{R}^{3\times3}$ and $\mathbf{T}_i^j \in \mathbb{R}^{3\times1}$.

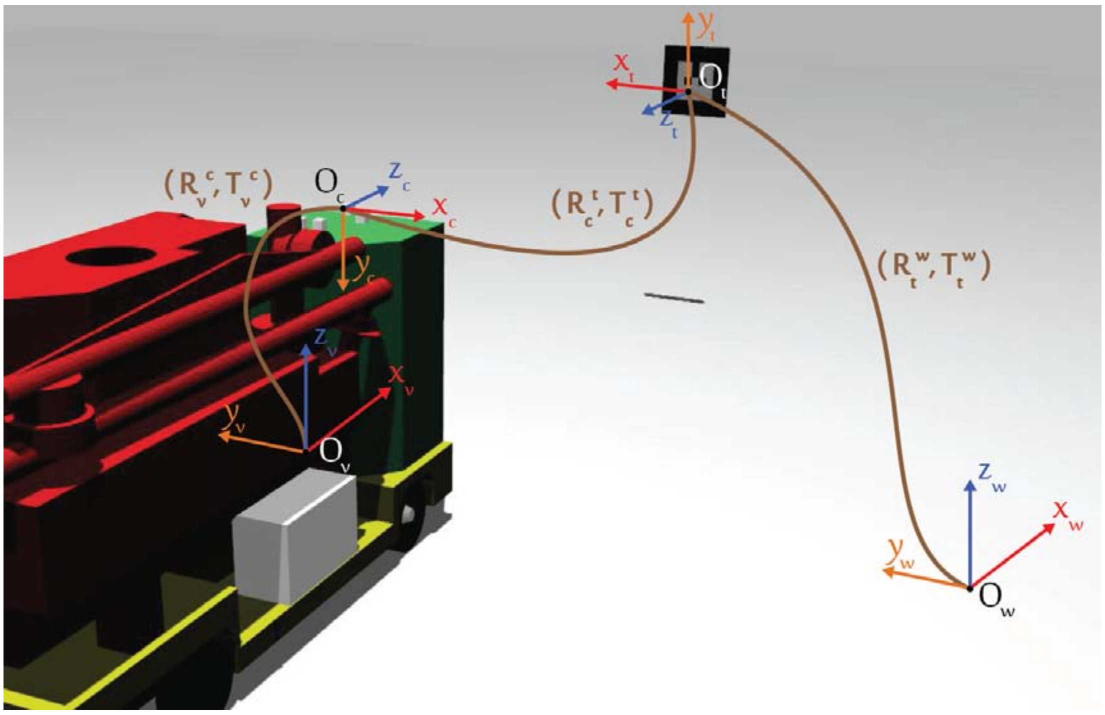


Figure 4.7: 3D reference frame of the vehicle, camera, tag and world

To clarify the following steps, a graphical definition of the tags, camera and vehicle reference frames is required: each of them is permanently attached to the relative object over the time and they are defined before the simulation starts. Figure 4.7 depicts the reference frames involved in the simulation: the red arrow corresponding to the x-axis, while the orange and blue ones are related, respectively, to the y-axis and z-axis. Frames are linked together by the rotations and translations defined above.

Vehicle pose with respect to the camera reference frame

Firstly, knowing the camera pose with respect to the vehicle frame, the vehicle pose with respect to the camera reference frame is easy to compute by post multiplying the translation vector $\mathbf{T}_c^v$ by the inverse of the rotation matrix $\mathbf{R}_c^v$ as follows:

$$\mathbf{T}_v^c = (\mathbf{R}_c^v)^{-1}\,\mathbf{T}_c^v = (\mathbf{R}_c^v)^T\,\mathbf{T}_c^v \tag{4.4}$$

$$\mathbf{R}_v^c = (\mathbf{R}_c^v)^{-1} = (\mathbf{R}_c^v)^T \tag{4.5}$$

Vehicle pose with respect to the tag reference frame

Once the vehicle pose with respect to the camera frame is computed, the main step of the vehicle indoor localization consists of retrieving the vehicle pose with respect to the tag reference frame using the information obtained by marker detection and localization process. Using the extrinsic and intrinsic parameters defined with the camera calibration, the pose of the i-th marker, meaning $\mathbf{R}_{t_i}^{c}$ and $\mathbf{T}_{t_i}^{c}$, is computed with respect to the camera frame. Then, the vehicle pose with respect to the tag reference frame is calculated by sequentially using the following equations:

$$\mathbf{T}_{v,t_i}^{c} = \mathbf{T}_{t_i}^{c} + \mathbf{T}_{v}^{c}, \qquad i = 0, \ldots, N-1 \tag{4.6}$$

$$\mathbf{T}_{v}^{t_i} = \left(\mathbf{R}_{t_i}^{c}\right)^{-1} \mathbf{T}_{v,t_i}^{c} = \left(\mathbf{R}_{t_i}^{c}\right)^{T} \mathbf{T}_{v,t_i}^{c}, \qquad i = 0, \ldots, N-1 \tag{4.7}$$

$$\mathbf{R}_{v}^{t_i} = \left(\mathbf{R}_{c}^{v}\right)^{-1} \left(\mathbf{R}_{t_i}^{c}\right)^{-1} = \left(\mathbf{R}_{c}^{v}\right)^{T} \left(\mathbf{R}_{t_i}^{c}\right)^{T}, \qquad i = 0, \ldots, N-1 \tag{4.8}$$

Vehicle pose with respect to the world reference frame

The final step of the vehicle localization is to retrieve the pose of the vehicle with respect to the world frame. The position of the FFV is computed taking into account the translation vectors $\mathbf{T}_{t_i}^{w}$, $i = 0, \ldots, N-1$, which are already known before the FFV initialization, while the orientation is performed by post multiplying the rotation matrices involved in the previous steps:

$$\mathbf{T}_{v}^{w} = \mathbf{T}_{t_i}^{w} + \mathbf{R}_{t_i}^{w} \mathbf{T}_{v}^{t_i}, \qquad i = 0, \ldots, N-1 \tag{4.9}$$

$$\mathbf{R}_{v}^{w} = \mathbf{R}_{t_i}^{w} \mathbf{R}_{c}^{t_i} \mathbf{R}_{v}^{c}, \qquad i = 0, \ldots, N-1 \tag{4.10}$$

Name and Description	***Algorithm Type***	***Input***	***Output***
RGB2gray (From RGB image to gray scale)	OpenCV function	RGB image	Gray scale image
Otsu Algorithm (Find optimal threshold)	OpenCV function	Gray scale image	Optimal threshold
Gray2Binary (from gray scale image to binary)	OpenCV function	Gray scale image - optimal threshold	Binary Image
Edge Detection (find image's edges)	OpenCV function	Binary image	Edges pixels
Quadrilateral Detection (find image's quadrilaterals)	OpenCV function	Edge pixels	Quadrilaterals Pixels
Rejection Criteria (select valid quadrilaterals)	Standard algorithm	Binary image - quadrilaterals pixels	Quadrilaterals pixels
Internal Area Sampling (retrieve bit sequences)	OpenCV function	Binary image - quadrilaterals pixels	Bit sequences
Forward Error Correction (detect and correct wrong bits)	Standard algorithm	Bit sequences	Bit sequences
Cyclical Redundancy Check (find valid markers)	Standard algorithm	Bit sequences - generator polynomial	Valid markers
Marker ID extraction (Id extraction from valid marker)	Standard algorithm	Bit sequence - valid markers	Marker ID
Corners Extraction (Corners extraction of valid markers)	OpenCV function	Binary image - quadrilateral pixels Valid Markers	Marker's corners
Vehicle pose w.r.t. camera frame	Standard algorithm	Camera pose w.r.t. vehicle frame	Vehicle pose w.r.t. camera frame
Vehicle pose w.r.t. marker frame	Standard Algorithm	Vehicle pose w.r.t. camera frame Marker's corners Intrinsic camera matrix	Vehicle pose w.r.t. marker frame
Vehicle pose w.r.t. world frame	Standard Algorithm	Vehicle pose w.r.t. marker frame Marker pose w.r.t. world frame	Vehicle pose w.r.t. world frame

Table 4.2: Indoor fiducial markers localization algorithms

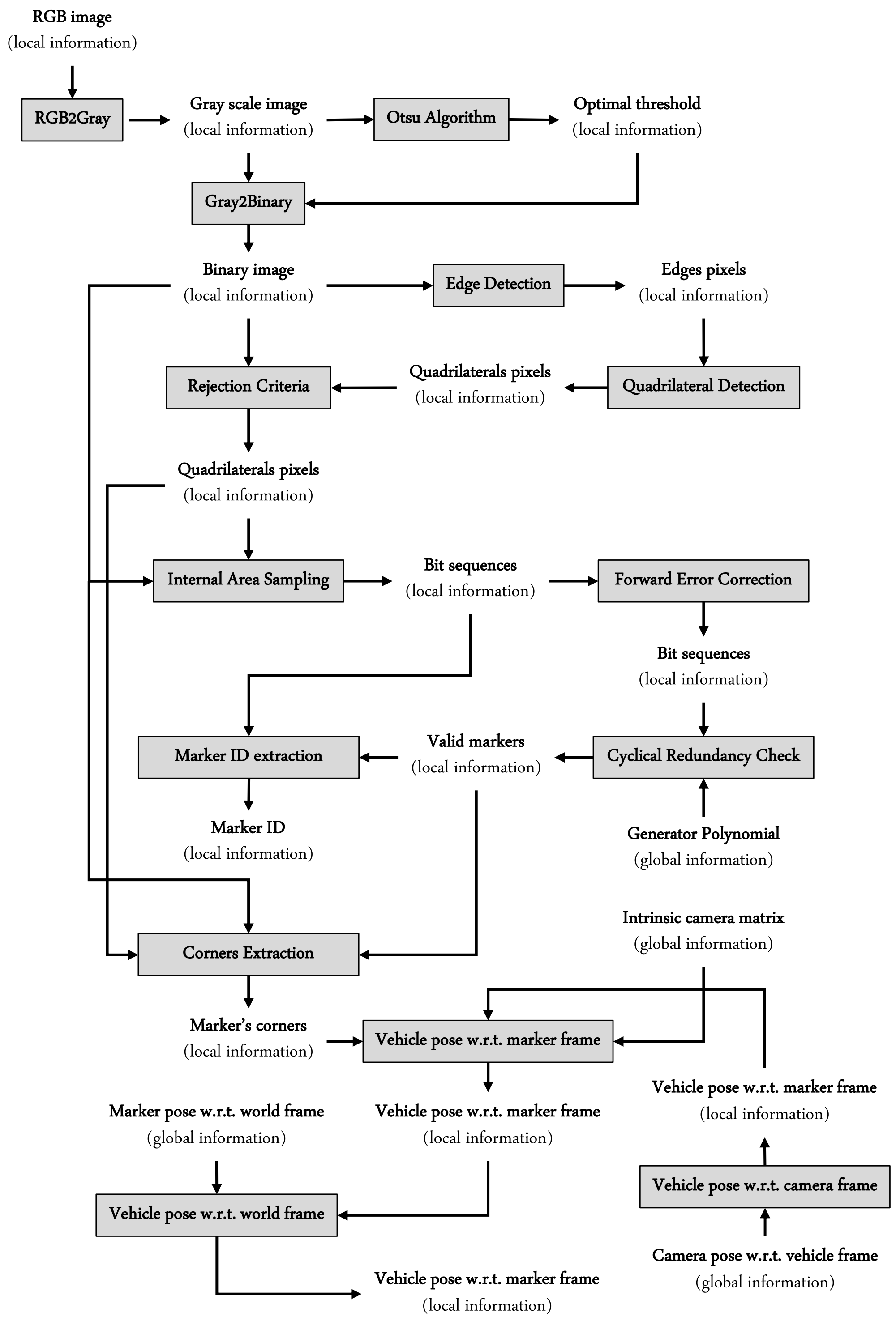


Figure 4.8: Indoor fiducial markers localization algorithms Flowchart

5
Wi-Fi Outdoor Localization

Nowadays, Global Position System (GPS) is the most widely used and the most successful positioning and localization technology. However, its low position accuracy and the high power consumption of the related sensors give poor performances in many localization applications where high accuracy is required. Moreover, many earth areas are not covered by GPS satellites or they are GPS denied. Hence, different kinds of outdoor localization technologies have been developed to support or to completely substitute the GPS technology in such scenarios with significant increment of the localization performances [22].
For this project, due to its flexibility, connectivity, mobility and low cost characteristics, the *Wireless* technology has been chosen for the outdoor localization process. To perform Wi-Fi localization it is necessary to define a wireless infrastructure called *Wireless Sensors Network* (WSN).
A WSN (Figure 5.1) is a radio-based self-configuring network consisting of a large number of wireless nodes equipped with sensing devices distributed in the environment where AGVs move. Each node is equipped with a transceiver to send and receive data with other nodes within its communication radio range. Sensed information in many applications of WSN

only becomes useful when it is accompanied by the location of the area and accurate distances of where such information is been sensed. Hence, sensor nodes need to know the distance between one another in order to calculate their positions[23].

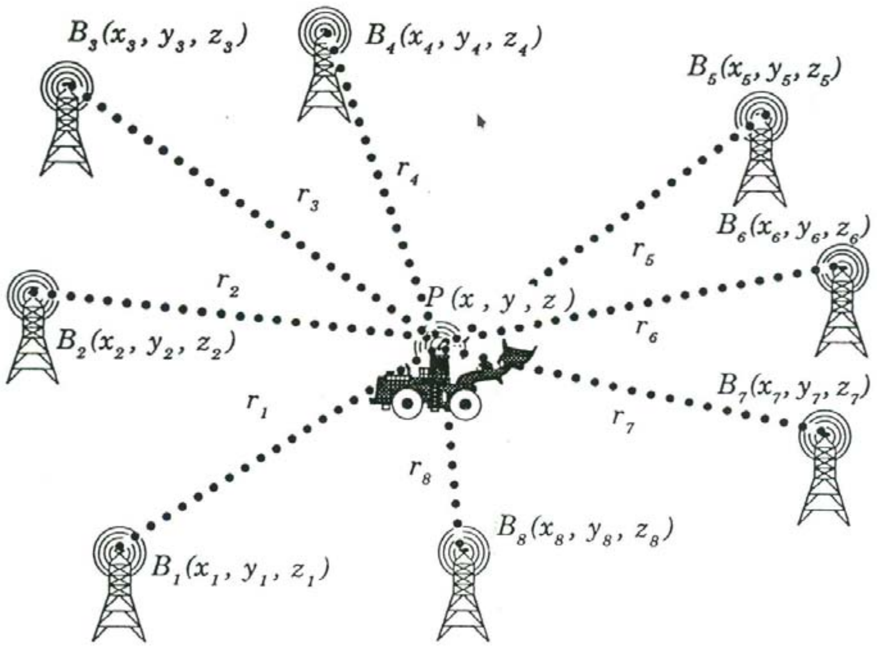


Figure 5.1: Wireless Sensors Network: a set of 8 beacons nodes $B_1 - B_8$ in known positions allows the localization of P

Right now, there are several techniques to compute position between wireless nodes, that are

- **Based on signal propagation speed and propagation time**
  - *Time of arrival (TOA)*
  - *Time of flight (TOF)*
  - *Time difference of arrival (TDOA)*

- **Based on signal direction**
  - *Angle of arrival (AOA)*

  - *Direction of arrival (DOA)*

- **Based on the power of the signal at the receiver**
  - *RSSI*

Due to its low cost, relative accuracy, presence in nearly almost all the wireless sensors and simple implementation on multiple simulation software, the RSSI technique is chosen for this project.

5.1 Received signal strength indicator

To compute the RSSI value, the Friis Transmission Equation is considered and implemented into a modified *Free Space Propagation Model*, which is one of the simplest propagation model for wireless transmission. In telecommunication, the Free Space Propagation Model assumes the transmitters and receivers wireless sensors placed in an empty environment with no absorbing obstacles or reflecting surfaces. However, in real world, wireless sensor networks infrastructure can be placed in messy environments such as cities or industrial plants. Indeed, wireless signal is affected by fixed and moving obstacles present in the surrounding environment, which can attenuate the power of the transmitted signal or add noise to the latter. To take into account these phenomena the *path loss exponent* (n) is defined. This coefficient assumes values between 2 (propagation in free space) and 6 (indoor environments). So, starting from the simplest Free Space Propagation Model, that is

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{c}{4\pi f d}\right)^2 = G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{5.1}$$

where

- $\mathbf{P_r}$ $[W]$ is the received signal power (RSSI)

- $\mathbf{P_t}$ $[W]$ is the transmitted signal power
- $\mathbf{G_t}$ is the transmitter gain
- $\mathbf{G_r}$ is the receiver gain
- **c** [*m/s*] is the light speed
- **f** [*Hz*] is the radio signal carrier frequency
- **d** [*m*] is the distance between transmitter and receiver
- $\boldsymbol{\lambda}$ [*m*] is the wavelength defined as $\frac{c}{f}$

the modified equation become

$$\frac{P_r}{P_t} = \frac{G_t G_r}{d^a}\left(\frac{c}{4\pi f}\right)^2 = \frac{G_t G_r}{d^a}\left(\frac{\lambda}{4\pi}\right)^2 \tag{5.2}$$

where **a** is the path loss exponent. Usually, in telecommunication, equations 5.1 and 5.2 are represented with powers and gains expressed in dBm (decibel milliwatt) and dBi (decibel isotropic) respectively

$$P(dBm) = 10\log_{10} P(W) \quad \text{and} \quad G(dBi) = 10\log_{10} G \tag{5.3}$$

Moreover the received signal, due to objects obstructing the propagation path between transmitter and receiver, can fluctuates: this effect is called *log-normal shadowing* and it is taken into account by adding an additive Gaussian white noise, $x \sim \mathcal{N}(0, \sigma^2)$, in the free space propagation model equation, that is

$$RSSI = P_r = P_t + G_t + G_r - x + 20\log_{10}\lambda - 20\log_{10}4\pi - 10\mathrm{a}\log_{10} d \tag{5.4}$$

where $RSSI$ is the estimated RSSI which takes into account both the path loss exponent and the log-normal shadowing noise. From equation 5.4, the estimated distance between each transmitter and receiver of the wireless sensor network can be computed as

$$d = 10^{\frac{P_t + G_t + G_r - RSSI + 20\log_{10}\lambda - 20\log_{10}4\pi}{10a}} \tag{5.5}$$

Let consider a WSN where multiple wireless transceiver are used to detect the position of an unknown point in the environment. The estimated distances between transceivers and receiver, which have been just retrieved using the equation 5.5, are used to compute the estimated position of the unknown point using a suitable localization approach. To accomplish this task, the *Trilateration* algorithm is chosen; its functionality and characteristics are described in the next section.

5.2 Trilateration algorithm

Trilateration technique is the process of determining the three dimensional coordinates x_p, y_p and z_p of an unknown point P by distance measurement, using the geometry of sphere; it does not involve angular measurements used, for example, in the triangulation process[24].

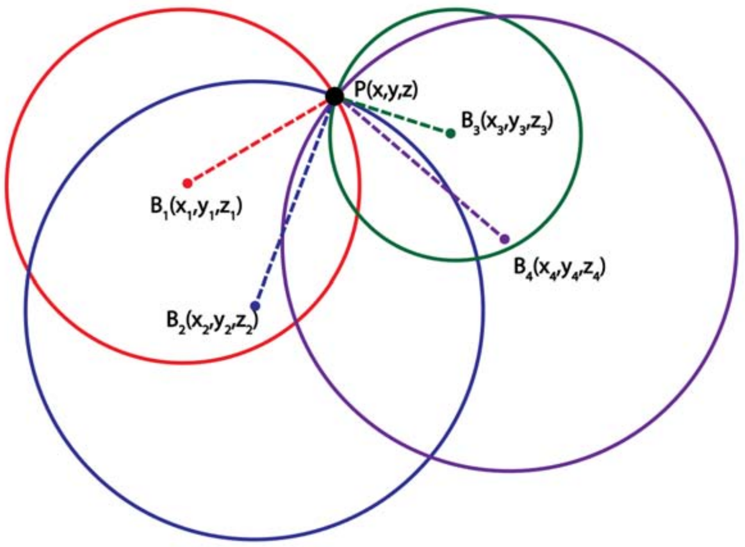


Figure 5.2: Trilateration technique: point P stands at the intersection of 4 spheres of center $B_1 - B_4$

Let consider a WSN with N fixed nodes (*beacons*) labeled B_i with pre defined 3D coordinates defined with respect to the world reference frame, that is

$$B_i = (x_i, y_i, z_i) \qquad i = 0, 1, 2, ...N - 1 \tag{5.6}$$

where i refers to the i-th beacon.
The trilateration algorithm considers the coordinates of the unknown point P as the point of intersection of several spheres, whose centers are the locations of the N beacons B_i (Figure 5.2). The equation of the N spheres are respectively

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r_i^2 \qquad i = 0, 1, 2, ...N - 1 \tag{5.7}$$

where r_i is both the radius of the i-th sphere and the true distance between the i-th beacon and the unknown point. The point of intersection of the $n < N$ spheres is obtained by letting $i = 0, 1, 2, ..., n - 1$ and solving the resulting n nonlinear system

$$\begin{cases} (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r_0^2 \\ (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = r_2^2 \\ \qquad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ (x - x_{n-1})^2 + (y - y_{n-1})^2 + (z - z_{n-1})^2 = r_{n-1}^2 \end{cases} \tag{5.8}$$

This approach is not feasible because it produces a nonlinear equation with high degree. Hence, another solving techniques needs to be take into account. Linearizing the system of equations geometrically converts the problem into one of finding the point of intersection of several planes (*direct approach*). When the exact distances from beacons are available, the solution of the linear system of equations is completely determined. Unfortunately, in real applications, distance between beacons and receiver, computed using the RSSI technique, is affected by noises. The estimated distance $\tilde{d}_i$ is obtained by adding error Δ_i to the true one d_i. Thus, in real cases, the position calculated by the direct solution of the linear equations is no longer acceptable meaning that the direct approach can not be used. The chosen approach consists of a first *linearization* step of the system 5.8 following by a *Linear Least Squares (LLS)* regression method. The latter is used to retrieve the estimated position of the unknown point in real cases where the exact distances between P and B_i are not available.

5.2.1 System linearization

Let consider the equation of *i*-th sphere with radius r_i, that is

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r_i^2 \tag{5.9}$$

The linearization technique uses a specific beacon B_j of coordinates x_j, y_j, z_j as linearizing tool. Adding and subtracting the B_j coordinates in 5.9 gives

$$(x - x_j + x_j - x_i)^2 + (y - y_j + y_j - y_i)^2 + (z - z_j + z_j - z_i)^2 = r_i^2 \tag{5.10}$$

where $i = 0, 1, 2, ..., j - 1, j + 1, ...n - 1$. Expanding and regrouping the terms, leads to

$$\begin{aligned}
&(x - x_j)(x_i - x_j) + (y - y_j)(y_i - y_j) + (z - z_j)(z_i - z_j) \\
&= \frac{1}{2}\left[(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2 - r_i^2 + (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right] \\
&= \frac{1}{2}\left[r_j^2 - r_i^2 + d_{i,j}^2\right] = b_{i,j}
\end{aligned} \tag{5.11}$$

where

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{5.12}$$

is the distance between beacons B_i and B_j. For simplicity, the beacon B_0 is selected as the specific one. Since $i = 1, 2, ...n - 1$, this leads to a linear system of $n - 2$ equations in 3 unknowns, that are the coordinates of the unknown point P. Therefore, at least 4 beacons are required to uniquely determine the position of P. Moreover, taking into account the

Wi-Fi RSSI values, only the *n* wireless transmitters for which the RSSI value retrieved at the receiver is over -85 [*dBm*] are taken into account for the trilateration algorithm. This improve the localization accuracy, avoiding noisy data or data loss. Indeed, -85 [*dBm*] is the minimum received power value for which the data can be received without information loss.

$$\begin{cases} (x-x_0)(x_1-x_0)+(y-y_0)(y_1-y_0)+(z-z_0)(z_1-z_0)=b_{1,0} \\ (x-x_0)(x_2-x_0)+(y-y_0)(y_2-y_0)+(z-z_0)(z_2-z_0)=b_{2,0} \\ (x-x_0)(x_3-x_0)+(y-y_0)(y_3-y_0)+(z-z_0)(z_3-z_0)=b_{3,0} \\ (x-x_0)(x_{n-1}-x_0)+(y-y_0)(y_{n-1}-y_0)+(z-z_0)(z_{n-1}-z_0)=b_{n-1,0} \end{cases} \tag{5.13}$$

This system can be easily written in matrix form

$$\mathbf{A}\rho = b \tag{5.14}$$

where

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} x_1-x_0 & y_1-y_0 & z_1-z_0 \\ x_2-x_0 & y_2-y_0 & z_2-z_0 \\ x_3-x_0 & y_3-y_0 & z_3-z_0 \\ \vdots & \vdots & \vdots \\ x_{n-1}-x_0 & y_{n-1}-y_0 & z_{n-1}-z_0 \end{bmatrix} \in \mathbb{R}^{n-1\times 3} \\ \rho &= \begin{bmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{bmatrix} \in \mathbb{R}^{3\times 1}, \quad b = \begin{bmatrix} b_{1,0} \\ b_{2,0} \\ b_{3,0} \\ \vdots \\ b_{n-1,0} \end{bmatrix} \in \mathbb{R}^{n-1\times 1} \end{aligned} \tag{5.15}$$

5.2.2 Linear least squares (LLS)

The linear least squares method is a standard approach to approximate the solution of ovedetermined systems (the number of equations is larger or equal to the number of unknowns) by minimizing the sum of the squares of the residuals made in the results of every single equation.
Consider the linear model defined in the equation 5.14. The least squares estimate of $\overrightarrow{x}$ is given by

$$\hat{\rho} = \arg \min_{\rho \in \mathbb{R}^{3\times 1}} \|b - \mathbf{A}\rho\|^2 \tag{5.16}$$

which is a quadratic form, convex in x. It means that all its global minimum points are given by setting its gradient to zero, that is

$$\begin{aligned} &\frac{\partial}{\partial \rho} \|b - \mathbf{A}\rho\|^2 \\ &\quad = \frac{\partial}{\partial \rho} (b - \mathbf{A}\rho)^T (b - \mathbf{A}\rho) \\ &\quad = \frac{\partial}{\partial \rho} \left(b^T b - b^T \mathbf{A}\rho - \rho^T \mathbf{A}^T b + \rho^T \mathbf{A}^T \mathbf{A}\rho\right) \\ &\quad = -2\mathbf{A}^T b + 2\mathbf{A}^T \mathbf{A}\rho = 0 \end{aligned} \tag{5.17}$$

At this point the linear least square estimator $\hat{x}$ can be easily computed as

$$\hat{\rho} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T b \tag{5.18}$$

which is well formed if and only if $\mathbf{A}^T\mathbf{A}$ is no singular or poorly conditioned. If $\mathbf{A}^T\mathbf{A}$ is rank deficient, even if the original matrix $\mathbf{A}$ was not close to singular, several approaches like **QR** or **SVD** decomposition can be take into account. The latter is used to compute the Moore-Penrose inverse matrix $\mathbf{A}^+$ of $\mathbf{A}$ to obtain the optimal solution of the system $\mathbf{A}\rho = b$, that is

$$\hat{\rho} = \mathbf{A}^+ b \tag{5.19}$$

Once the least square estimator is found, the final step is to retrieve the coordinates of the wireless receiver P in the 3D space. Indeed, the coordinates of the specific beacon B_0 used to linearize the system are added to $\hat{x}$.

$$P = \hat{\rho} + B_0 \Longrightarrow \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \hat{x} - x_0 \\ \hat{y} - y_0 \\ \hat{z} - z_0 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{5.20}$$

Trilateration algorithm is used, most of the times, for localization purposes meaning both position and orientation of the object, the fluoride feeder vehicle, can be computed. Unfortunately, because of the trilateration process not use angular measurement but only distance information between beacons and unknown point, only the position can be computed.
To determine the orientation of the vehicle, it is necessary to merge the position information obtained via trilateration algorithm with the ones obtained using inertial measurement unit sensors (gyroscopes, accelerometers, etc.) utilizing suitable sensor fusion approach (Chapter 6). Finally, notice that, the vector P contains the 3D coordinates of the wireless receiver with respect to the world frame. To easily compute the estimated position, receiver P needs to be placed on the top of the vehicle (for example, over the cabin) and centered on the origin of the latter coordinate frame system. Once the coordinates of the receiver are computed, the ones of the vehicle can be determined subtracting the translation vector which link the receiver P to the center of the vehicle coordinate frame.

$$\mathbf{V} = P - \mathbf{T_r^v} \Longrightarrow \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} - \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \tag{5.21}$$

6 Sensors Fusion

In real world, vehicle localization is affected by various environmental noises, interferences and conditions that make this task no trivial. For example, wireless signals can be occluded by obstacles or reflected by building walls causing wrong RSSI detection and low accuracy outdoor localization. Moreover, markers detection can be affected by camera non-idealities. These interferences act as error sources, hence preventing the FFV to know its position and orientation with precision and accuracy. FFV perceives the environment using sensors, which enable the vehicle to receive data about the surrounding area and are used to determine its own pose with respect to the 3D world coordinate frame. In particular, the FFV is equipped with wireless communication system, camera, laser and inertial measurement sensors (Section 7.3.2). Because these sensors directly interact with the surroundings, the noises impact the data reading and acquisition. So, it is preferable to use multiple sensors in order to increase the redundancy of the system, hence improving the probability of getting accurate data. To fuse the data acquired by the sensors mounted on the FFV, the *Extended Kalman filter (EKF)* sensors fusion technique is used [25].
EKF is chosen with respect to the classic Kalman filter to deal with nonlinear systems. The

requirement of linear equations for the measurement and state transition models is, thus, relaxed; indeed, the models can be nonlinear and need only to be differentiable. The EKF works by transforming the nonlinear models at each time step into linearized ones.

6.1 Extended Kalman Filter Framework

Firstly, let consider the non-linear continuous-time dynamic system of the EKF, that is

$$\begin{cases} \dot{\mathbf{x}}(t) & = f(\mathbf{x}(t)) + \mathbf{w}(t) \\ \mathbf{z}(t) & = h(\mathbf{x}(t)) + \mathbf{e}(t) \end{cases} \tag{6.1}$$

where

- $\mathbf{x}(\cdot)$ is the $n\times 1$ state vector at time *k*
- $\mathbf{f}(\cdot)$ is the $n\times$n process nonlinear vector function
- $\mathbf{w}(\cdot)$ is the $n\times 1$ process noise vector
- $\mathbf{z}(\cdot)$ is the $m\times 1$ observation vector
- $\mathbf{h}(\cdot)$ is the $m\times 1$ observation nonlinear vector function
- $\mathbf{e}(\cdot)$ is the $m\times 1$ measurement noise vector

In particular, the noise processes are zero-mean random vectors with known covariances, uncorrelated each other and temporally uncorrelated with the initial state value $\mathbf{x}_0$:

$$\begin{gathered} \mathbf{w} \in \mathbb{R}^{n\times 1},\ E[\mathbf{w}] = 0,\ Var[\mathbf{w}] = E[\mathbf{w}\mathbf{w}^T] = Q = Q^T \geq 0 \in \mathbb{R}^{n\times n} \\ \mathbf{e} \in \mathbb{R}^{m\times 1},\ E[\mathbf{e}] = 0,\ Var[\mathbf{e}] = E[\mathbf{e}\mathbf{e}^T] = R = R^T > 0 \in \mathbb{R}^{m\times m} \\ E[\mathbf{w}\mathbf{e}^T] = 0, \quad E[\mathbf{w}\mathbf{x}_0^T] = 0, \quad E[\mathbf{e}\mathbf{x}_0^T] = 0 \end{gathered} \tag{6.2}$$

$$F(\overline{\mathbf{x}}(t)) = \frac{\partial f(\mathbf{x}(t))}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\overline{\mathbf{x}}(t)} \qquad H(\overline{\mathbf{x}}(t)) = \frac{\partial h(\mathbf{x}(t))}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\overline{\mathbf{x}}(t)} \tag{6.3}$$

The EKF works in two steps:

- *Prediction* in which it is carried out a prediction step that projects the current state estimate and error covariance forward in time
- *Update* in which the current a priori prediction is combined with current observation information to refine the state estimate

The *initialization* step, in which vectors and matrices of the dynamic system are initialized, is the first fundamental step to perform before the EKF started. Vectors and matrices are set with values obtained with the first FFV sensors measurements data.

6.2 Extended Kalman Filter Prediction Step

Assuming that the dynamic of the vehicle is slower than the sampling time

$$\Delta t_k = t_k - t_{k-1} \tag{6.4}$$

It is possible to calculate both a priori state and error covariance using the *exact discretization*:

$$\hat{\mathbf{x}}(k|k-1) = f_{k-1}(\mathbf{x}(k-1|k-1)) \tag{6.5}$$

$$\hat{P}(k|k-1) = F(k-1|k-1)P(k-1|k-1)F^T(k-1|k-1) + Q(k-1) \tag{6.6}$$

where$f$ is the standard 3D kinematic model derived from Newtonian mechanics.

6.3 Extended Kalman Filter Update Step

Once the computation of the a priori state and error covariance is done, the a posteriori state estimation and error covariance $P \in \mathbb{R}^{n\times n}$ can be defined using the following relations:

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + L(k)[\mathbf{z}(k) - H(k-1|k-1)\hat{\mathbf{x}}(k|k-1)] \tag{6.7}$$

$$P(k|k) = [I - L(k)H(k|k-1)]\hat{P}(k|k-1)[I - L(k)H(k|k-1)]^T + L(k)RL(k)^T \quad (6.8)$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix and $L \in \mathbb{R}^{n \times n}$ is the *Kalman gain* calculated as follows:

$$L(k) = \hat{P}(k|k-1)H^T(k|k-1)[H(k|k-1)P(k|k-1)H^T(k|k-1) + R(k)]^{-1} \quad (6.9)$$

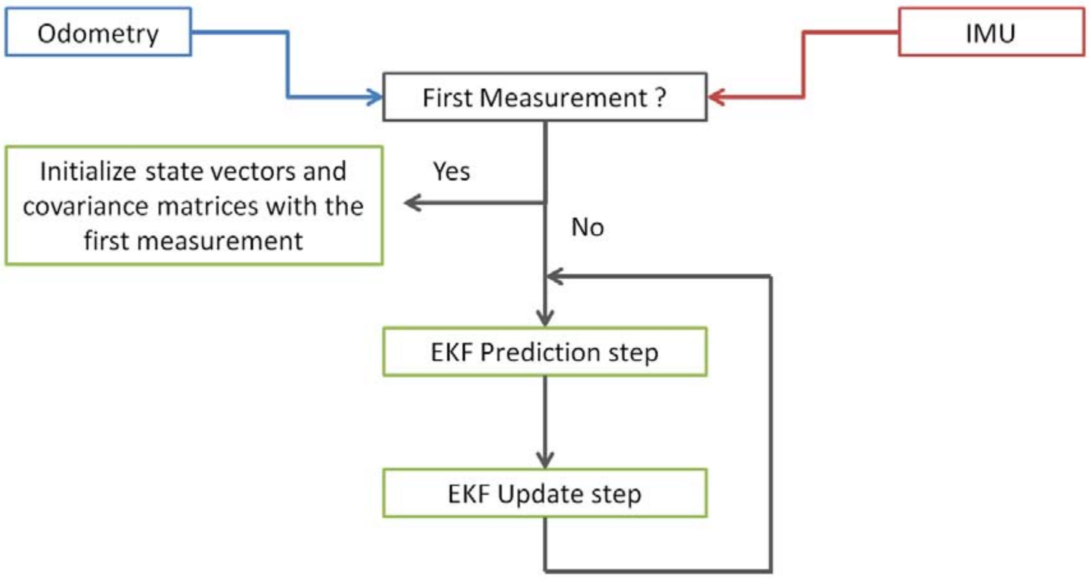

Odometry
IMU
First Measurement ?
Yes
Initialize state vectors and covariance matrices with the first measurement
No
EKF Prediction step
EKF Update step

Figure 6.1: Sensors fusion technique

6.4 Sensors Fusion using Odometry and Inertial information

This section describes a model for sensors fusion using the odometry and the inertial information retrieved by the sensors attached to the FFV. To describe the model, the fiducial markers indoor localization technique is chosen. Indeed, with the indoor localization, the 3D full pose (position + orientation) can be retrieved. Similar model can be applied on the wireless outdoor localization where the orientation is not retrieve by the trilateration algorithm (Section 5.2). In the latter, using only the Inertial Measurement Unit (IMU) sensor, is possible to retrieve only the *relative* orientation of the FFV with respect to its starting orientation defined with respect to the world reference frame.

6.4.1 Inertial Measurement Unit Sensors

The IMU sensors are based on inertia and relevant measuring principles. IMU for aerial and ground robots and vehicles typically consists of *accelerometers*, *gyroscopes* and sometimes also *magnetometers*. Subsequently, a briefly description of the main characteristics of accelerometers and gyroscopes widely used are depicted.

- *Accelerometers*: they are electromechanical devices that are able of measuring static and/or dynamic forces of acceleration. Static forces include gravity, while dynamic forces can include vibrations and movement. Accelerometers can measure acceleration on 1, 2 or 3 axes.

- *Gyroscopes*: a gyroscope is, conceptually, a spinning wheel in which the axis of rotation is free to assume any possible orientation. When rotating, the orientation of this axis remains unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum. Due to this principle, a gyroscope can lead to the measurement of orientation and its rate of change.

Real inertial measurement sensors are affected by noises: accelerometers, even when there are no movements performed by the vehicle, produce a small time-varying offset (*bias* error), while gyroscopes measurements are affected by noise. Considering the output orientation data, these two errors can cause both measurement error and the orientation *drift* phenomenon. Sensors fusion technique is used to compensate measurement noise errors, while bias offset is reduced performing a closed path in which IMU output data are recalibrated in order to connect the starting point of the path with the ending one. In this project, an *high precision pre initialized IMU* is considered, meaning only the noise produced by the gyroscope is considered, while the bias produced by the accelerometers is considered negligible.

6.4.2 Model Equations

Let

$$\mathbf{x} = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma & v_X & v_y & v_z & \omega_R & \omega_P & \omega_Y & a_x & a_y & a_z \end{bmatrix}^T \in \mathbb{R}^{15\times 1} \tag{6.10}$$

be the state vector where

1. x, y and z are the 3D position coordinates ([*m*])
2. α (roll), β (pitch) and γ (yaw) are the 3D orientation coordinate ([*rad*])
3. v_x, v_y and v_z are the linear velocities ([*m/s*])
4. ω_R, ω_P and ω_Y are the angular velocities ([*rad/s*])
5. a_x, a_y and a_z are the linear accelerations ($[m/s^2]$)

In particular, the 3D full pose is obtained by using the odometry information (visual camera and wireless sensors), while the angular velocities and linear accelerations elements are obtained using the IMU information (IMU sensor). Moreover, no information about linear velocities are retrieved by both the vehicle odometry and inertial sensors.
In order to clarify the dynamic model definition, let us split the state vector $\mathbf{x}$ into five vectors:

1. $\mathbf{p} = [x, y, z]^T \in \mathbb{R}^{3\times 1}$: position vector w.r.t. the world frame
2. $\boldsymbol{\theta} = [\alpha, \beta, \gamma]^T \in \mathbb{R}^{3\times 1}$: orientation vector w.r.t. the world frame

3. $\mathbf{v} = [v_x, v_y, v_z]^T \in \mathbb{R}^{3\times1}$: linear velocities vector w.r.t. the vehicle frame

4. $\boldsymbol{\omega} = [\omega_R, \omega_P, \omega_Y]^T \in \mathbb{R}^{3\times1}$: angular velocities vector w.r.t. the vehicle frame

5. $\mathbf{a} = [a_x, a_y, a_z]^T \in \mathbb{R}^{3\times1}$: linear accelerations w.r.t. the vehicle frame

such that $\mathbf{x} = \begin{bmatrix} \mathbf{p} & \boldsymbol{\theta} & \mathbf{v} & \boldsymbol{\omega} & \mathbf{a} \end{bmatrix}^T$.

The dynamic model system is defined as follows:

$$\dot{\mathbf{x}} = \begin{cases} \dot{\mathbf{p}} = \mathbf{p} + R^w_{v,l}\,\mathbf{v}dt + \frac{1}{2}R^w_{v,l}\,\mathbf{a}dt^2 \\ \dot{\boldsymbol{\theta}} = \boldsymbol{\theta} + R^w_{v,a}\,\boldsymbol{\omega}dt \\ \dot{\mathbf{v}} = \mathbf{v} + \mathbf{a}dt \\ \dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}} \\ \mathbf{a} = \mathbf{a} \end{cases} \tag{6.11}$$

$$\mathbf{z} = h(\mathbf{x}) + \mathbf{e} = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma & \omega_R & \omega_P & \omega_Y & a_x & a_y & a_z \end{bmatrix}^T + \mathbf{e}$$

where the matrix $R^w_{v,l}$ and $R^w_{v,a}$ are 3×3 rotation matrices which transform the linear and angular velocities defined on the vehicle reference frame in the 3D world coordinates system.

7 Simulation setup

In this chapter, the crucial step of simulation setup is described. Firstly, a brief overview of the hardware specifications and all the software used to perform robust and efficient experiments are depicted, highlighting the ones used to run the simulations. Then, the processes of 3D modeling of the FFV and the aluminum smelter indoor and outdoor environment are given. Finally, the virtual sensors attached to the FFV frame are analyzed describing all their fundamental parameters.

7.1 Hardware and Software Setup

In this section, a briefly overview of hardware (PC specification) and software used for simulation is made. The latter one can be divided in:

- **Pre-Simulation** software used to create the simulation setup.
- **Simulation** software used to run the simulation.
- **Post-Simulation** software used to data acquisition and elaboration.

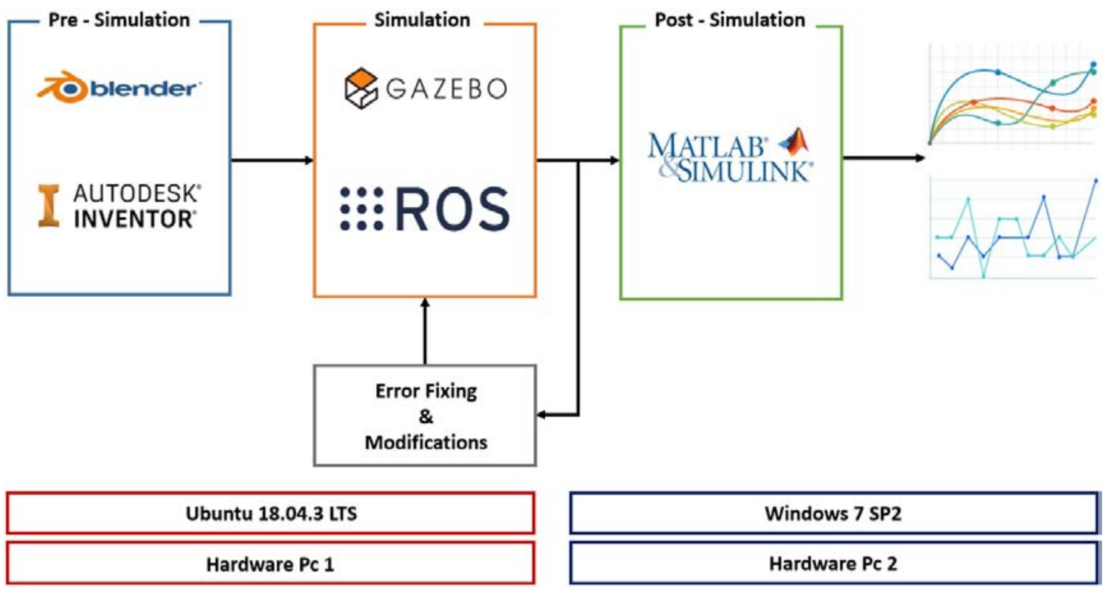


Figure 7.1: Simulation setup

7.1.1 PC Specification

Due to the computational burden of the simulation and because of logical separation in the simulation processes, a two PC setup is chosen for the simulation. The first one, the *main PC*, is used to run the simulation software (ROS and Gazebo) while, the second one is used to acquire data from the simulation (Matlab and Simulink) or to modify simulation setup using a suitable editor (Table 7.1).

PC-1 Main Specifications	
CPU	Intel i7-2630QM
RAM	4 Gb
GPU	Nvidia GT540M
O.S.	Ubuntu 18.04.3 LTS

PC-2 Main Specifications	
CPU	Intel i3-6006U
RAM	4Gb
GPU	Intel HD 3000
O.S.	Windows 7 SP2

Table 7.1: PC main specifications

7.1.2 Blender

Blender is a free, multi-platform and open-source 3D software used for animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. For the project, Blender has been used to model the 3D indoor and outdoor world model (Section 7.2) and to create the 3D graphic meshes used for Fluoride Feeder vehicle, ARTag markers and Wi-Fi transceivers.

Figure 7.2: Blender logo

7.1.3 Autodesk Inventor

Autodesk Inventor is a 3D mechanical solid modeling design software developed by Autodesk. It is used for 3D mechanical design, design communication, tooling creation and product simulation. This software enables users to produce accurate 3D models to aid in designing, visualizing and simulating products before they are built. For the thesis, Autodesk Inventor has been used to convert the Fluoride Feeder vehicle and potcels 3D Inventor proprietary files (*.iam* and *.ipt*) into a 3D ones supported by Blender (*.dae*). Furthermore, information about mass and inertial of FFV and potcel have been retrieved and used for the Gazebo Simulation.

Figure 7.3: Inventor logo

7.1.4 Robotic Operative System (ROS)

The Robot Operating System (ROS) is a flexible framework for writing robot software which runs on Unix-based platforms. It provides services expected from operating system such as hardware abstraction, low level device control, message passing between processes and package management. It also includes tools, libraries, and conventions that aim

Figure 7.4: ROS logo

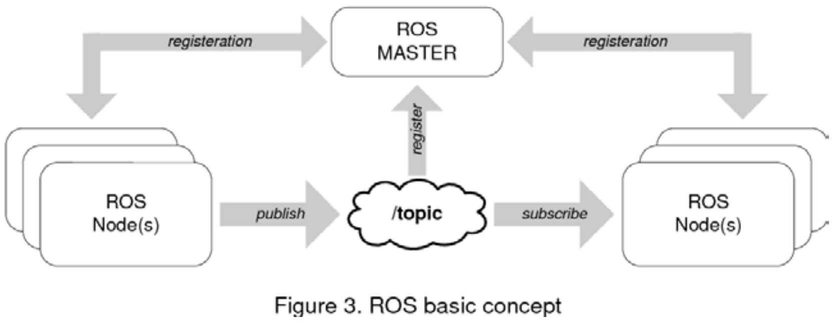


Figure 7.5: ROS framework

to create and simulate complex and robust robot behavior (robot arms, vehicles, drones, ...) across a wide variety of robotic platforms. Moreover, due to the ROS packages organization and combinations, it can be used for collaborative projects and works. The graph-based framework uses *nodes* to identify processes which can send (publish) or receive (subscribe) data through abstract communication channels referred to as *topics*. Topic is a bus which carries messages of any type, such as sensors information and data acquisition or control input values, in anonymous way: this means that nodes are not aware of whom they are communicating with. Nodes that send data over topics are called *publisher*, while the one that receive data from topics are called *subscriber*. *Network management* is performed by a master node called using the *roscore*. Roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system. and it has to run in order for ROS nodes to communicate. Example of topics data type used in this project are:

nav_msgs/Odometry: represents an estimate of a pose (position and orientation) and velocity (linear and angular) in free space. The pose in this message should be specified in the coordinate frame given by *header.frame_id*. The twist in this message should be specified in the coordinate frame given by the *child_frame_id*.

sensor_msgs/ImageRaw: define information about frames acquired by a given camera. Description about camera resolution and, thus, image size are given. Finally information about image pixels are stored in the *data* matrix of size equal to the image pixels dimensions.

sensor_msgs/CameraInfo: defines meta information of the camera. In particular, information about image size, intrinsic, extrinsic and projection camera matrices and distortion parameters are given. These data are used in the marker localization technique (Chapter 4) to retrieve tags position with respect to the camera frame.

sensor_msgs/LaserScan: represents data of laser measures and laser information. In particular, information about minimum, maximum and increment angle of scanning and range of laser operation distances are given. Laser measures, which are distances between laser and objects for each angle bin, are stored in a *range* array which length is given by the equation

$$\text{length(range)} = \frac{\alpha_M - \alpha_m}{\alpha_i} \tag{7.1}$$

where α_M and α_m are, respectively the maximum and minimum scanning angle, while α_i is the angle increment. Keep in mind that, increasing the angle increment of the laser scanner gives better speed performances but lower measure accuracy, while the vice versa gives lower speed performances but high measure accuracy.

sensor_msgs/Imu: define information about orientation, angular velocity and linear acceleration of the vehicle, obtained by the inertial measurement unit.

7.1.5 Gazebo Simulator

Gazebo Simulator is an open-source 3D robotics simulator for Linux operative systems. Used in combination with ROS (Subsection 7.1.4), it's possible to rapidly test algorithms, design robots, maps robots movements and run simulation in different scenarios. Moreover, due to its robust physics engine, offers the ability to accurately and efficiently simulate populations of robots both in indoor

Figure 7.6: Gazebo logo

and outdoor environments. For the project, Gazebo Simulator has been used to test the simulation and acquire the needed information for real-time user algorithms.
A Gazebo model is defined using the *Unified Robot Description Format* (URDF), an XML format which uses *specific tags* to describe kinematics, dynamics and basic physics description of a robot. General robot is defined in the URDF format using the following tags:

Link: describes the rigid body of a robot element with inertia, visual features, and collision properties (Figure 7.7).

- *Visual*: it defines the visual properties of a link. This element specifies shape (box, cylinder, etc.), dimensions (width, height, thickness, etc.) and color of the object. Shape of the link can be both geometrical or a complex shape given by 3D mesh.
- *Collision*: the collision element is a direct sub element of the link object and it defines the collision properties of a link. Even if, most of the times it is equal to the visual element, in order to lighten the simulation, the geometry shape is simpler than the visual one (for example, a bounding box).
- *Inertial*: it defines the inertial properties of a link, such as center of mass and inertial matrix.

```xml
<link name="my_link">
   <visual>
      <origin> ... </origin>
      <geometry> ... </geometry>
   </visual>
   <collision>
      <origin> ... </origin>
      <geometry> ... </geometry>
   </collision>
   <inertial>
      <mass> ... </mass>
```

```
            <inertia> ... </inertia>
            <axis> ... </axis>
        </inertial>
    </link>
```

Listing 7.1: Template of XML format for Link element

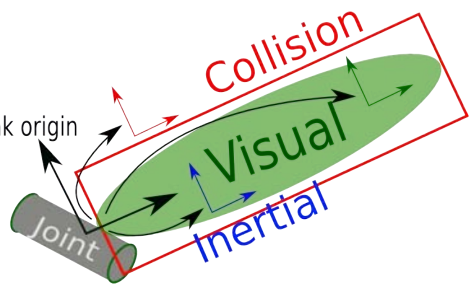


Figure 7.7: Link description

Joint: describes the kinematics and dynamics of the joint (Figure 7.8). Moreover, due to hard constraints and physics limitation of the robot (maximum steering angle, maximum velocity, etc.), joint can specifies the so called *safety limits*.

- *Type*: specifies the type of joint
  - *Revolute*: a hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits
  - *Continuous*: a continuous hinge joint that rotates around the axis and has no upper and lower limits
  - *Prismatic*: a sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits
  - *Fixed*: this is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint does not require the axis, calibration, dynamics, limits or safety controller
  - *Floating*: this joint allows motion for all 6 degrees of freedom

  - *Planar*: this joint allows motion in a plane perpendicular to the axis

- *Parent*: specifies the name of the parent link in the robot tree structure
- *Child*: specifies the name of the child link in the robot tree structure w.r.t. the parent link
- *Axis*: specifies the joint axis specified in the joint frame. This is, for example, the rotation axis for revolute joint or the translation axis for the prismatic joint
- *Limit*: specifies the lower and upper joint limit, the maximum effort and velocity of the joint.

```xml
<joint name="joint_name" type="...">
   <origin xyz="..." rpy="..."/>
   <parent link="parent_link"/>
   <child link="child_link"/>
   <limit effort=".." velocity=".." lower=".." upper=".."/>
</joint>
```

Listing 7.2: Template of XML format for Joint element

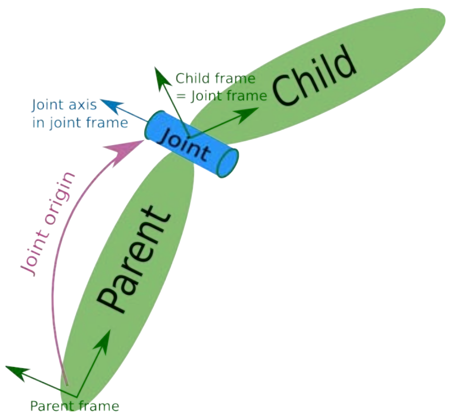


Figure 7.8: Joint description

Finally, the gazebo simulation world and the robot itself can be equipped with different types of controls and sensors. Controls allow the robot to move in all the possible direction, while sensors guarantee interaction between robot and the surrounding environment: for example, robot is capable to detect and avoid obstacles, to move along a given path to perform tasks, localize itself or to compute its linear and angular velocities. Controls and sensors, in order to work properly inside the Gazebo simulation, need to be attached to the so called *Gazebo Plugins*. Gazebo Plugin is a chunk of code, usually written in C++, which is compiled as a shared library and inserted into the simulation. There are currently 6 types of plugins: World (control world properties such as physics engine, ambient lighting, etc.), Model (control joints, and model state), Sensor (acquire sensor information and control sensor properties), System, Visual and GUI plugins. To apply sensors to the related robot, two steps needs to be performed:

1. Create a link related to each sensor: for increase the simulation reality, link dimensions and shapes should be as similar as possible to a real sensor

2. Attach the plugin to the sensor link: to attach the plugin to the sensor link, specific tags are required (Listing 7.3):

    - *Gazebo*: is an extension of the URDF used for specifying additional properties needed for simulation purposes in Gazebo
    - *Sensor*: specifies the type of sensor and its name. Inside sensor tag, sensor properties are defined
    - *Plugin*: specifies the name and the filename of the gazebo sensor plugin. Inside plugin tag, ROS communication properties, such as topic and frame name, are depicted. Plugin only-reading file has extension *.so* and it is created when the plugin is compiled.

```xml
<gazebo reference="...">
   <sensor type="..." name="...">
      <visualize>true</visualize>
      <update_rate>15</update_rate>
      <ray>
         <scan...</scan>
         <range...</range>
         <noise...</noise>
      </ray>
      <plugin name="..." filename="...">
         <topicName...</topicName>
         <frameName...</frameName>
      </plugin>
   </sensor>
</gazebo>
```

Listing 7.3: Template of XML format for laser scanner sensor plugin element

7.1.6 Matlab and Simulink

Matlab is a program designed specifically for engineers and scientists. Using Matlab you can analyze data, plot different types of graphs, develop algorithms, create models and applications. Simulink is a block diagram environment for simulation and model-based design. It supports system-level design, simulation, automatic code generation and ROS integration. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with Matlab, enabling the users to incorporate Matlab algorithms into models and export simulation results to Matlab for further analysis.For the project, Matlab

MATLAB® &SIMULINK®

Figure 7.9: Matlab and Simulink logo

and Simulink are used to define end-user plots to better understand how the simulations works in terms of mean, variance and standard deviation of position and orientation error.

7.2 World Modelling

In order to perform Gazebo simulations so that they are as realistic as possible it is necessary to define a model for the 3D world. The latter represents the environment where the vehicle can move and complete the tasks assigned to it. In particular, for the purpose of this project, a simplified version of an aluminum production plant has been adopted, as depicted in Figure 7.10.

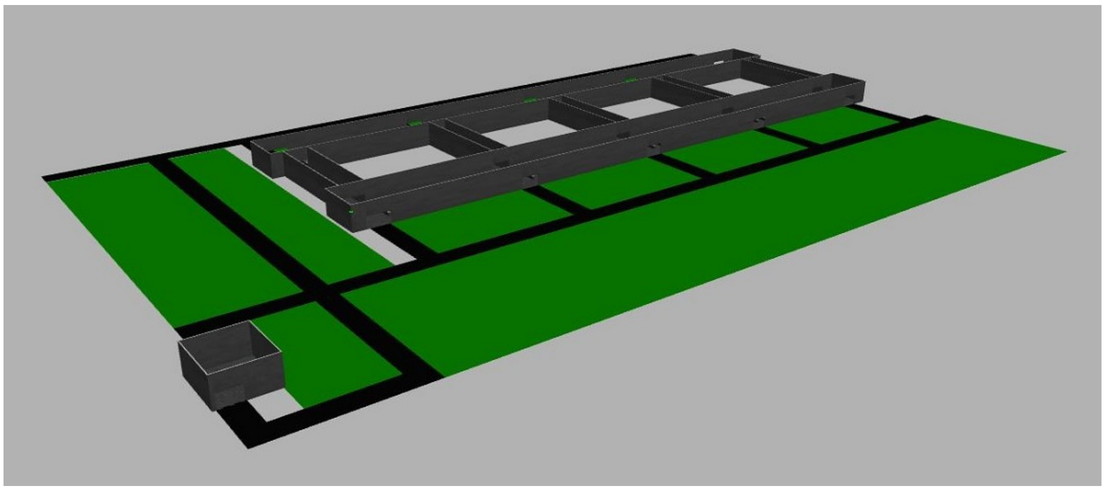


Figure 7.10: View of the simplified world model

Notice that this choice does not affect the reliability of the simulation and allows to reduce the computational burden required by the latter.
The 3D world just introduced can be described using three main elements:

- *Potroom*: already introduced in Section 2.3.1, it is the building where the aluminium production process is performed. Each of the potcels contained within it could represent the final point for the path that the vehicle has to follow;
- *Aluminium Fluoride Storage and Handling* (AFSH): it is the building where the vehicle's tank is filled with the aluminium fluoride, which will then have to be distributed

in the potcels. According to the given tasks schedule, the AFSH may represents the starting or an intermediate area through which the vehicle's path has to pass;

- *Streets*: they define the areas that the vehicle can cover to move from one building to an other. In general the points of the streets constitute the greater part of those that describe the vehicle's path.

In the following sections, a brief description of how these three elements have been implemented is given.

7.2.1 Potrooms and AFSH

Potrooms

In general, a Potroom can be described as a rectangular building that contains a sequence of Potcels, placed in series or parallel, and one or more hallways that allows vehicles and human operators to easily move from one point to an other inside the room. Additionally a crane could be present to move heavy loads. Since for the accomplishment of the objectives of this analysis such device is not influential, it has been discarded in the Potroom modelling phase. For the same reason, also the design of the building's roof has been neglected.
At this point, in order to obtain a 3D representation of the Potroom, the 3D computer graphics software Blender (Section 7.1.2) has been adopted. The design procedure can be divided in three phases:

- *Potcel design:* first a model for a potcel needs to be obtained. To this end, a 3D rendering of the object, kindly provided by Techmo s.p.a., was imported into the aforementioned software. The potcel's local reference frame has then been moved in order to make the positioning of the object inside the building easier. Furthermore a metal texture has been added in order to get a more realistic visualization of the model. The result of this step can be seen in Figure 7.11.

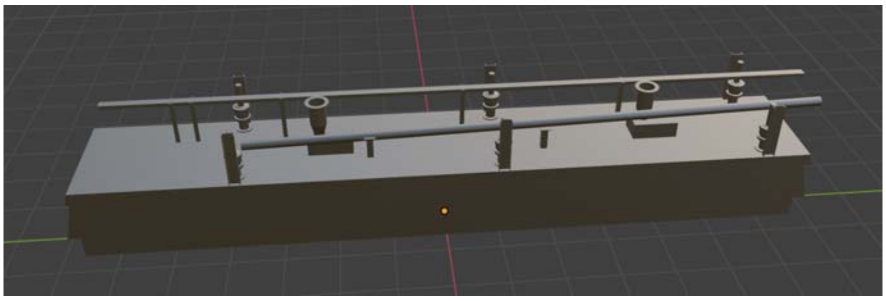


Figure 7.11: View of the Potcel model

- *Room design:* in this phase it is desired to get a model for the building. To this aim it is necessary to define the floor and the walls structure. Regarding the second, it has been obtained as the union of parallelepipeds of different dimensions. A part of these is in contact with the ground while another part is not, in order to simulate the shape of the doors entering and leaving the room. In order to get a more realistic visual of the result a texture similar to the one of a concrete wall has been added to each defined element. For what concern the floor it has been simply defined using a rectangular plane with the same dimensions of the building plan. Then an asphalt texture has been added to the floor model in order to obtain a better visual result. The room obtained from this procedure can be seen in Figure 7.12.

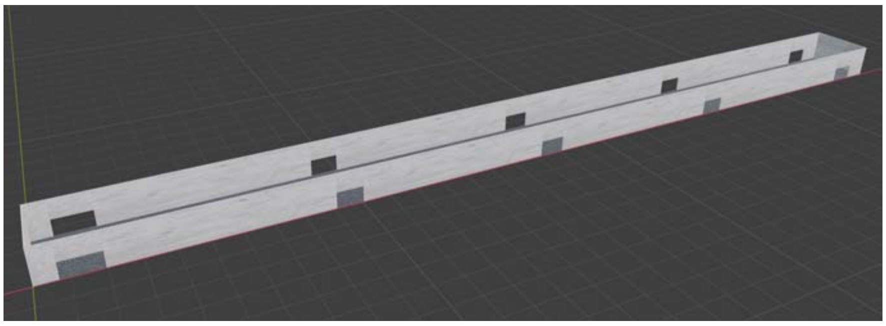


Figure 7.12: View of the room model

- *Potroom design:* in this last phase the results of the two previous steps have been added in order to obtain the final 3D model of the potroom. In particular 40 potcels have been inserted in parallel in the room model. This arrangement allows to obtain an area that can serve as a large hallway. In order to get a more rich simulation it has been decided to include in the world representation a second Potroom, identical and connected to the first one through two long corridors. The latters have been implemented using the same procedure used to define the room during the second step. To perform indoor localization, ARTag markers are placed on the potroom wall with position and orientation with respect to the world frame defined in Table A.5 and Table A.6. In total, 29 tags are placed on the potrooms: 9 are placed on the long side walls of the first potroom, 12 are placed on the long side walls of the center corridor which links the two potrooms and 8 are placed on the long side walls of the second potroom. Each tag is created starting with an ARTag marker image: the latter is used as texture in Blender in order to create the tag mesh (*.dae* file) for the related *.urdf* file. For this project, due to the high dimensions of the potroom building, tags dimensions are 1x1 $[m^2]$. The final 3D model of the potrooms without markers is depicted in Figure 7.13.

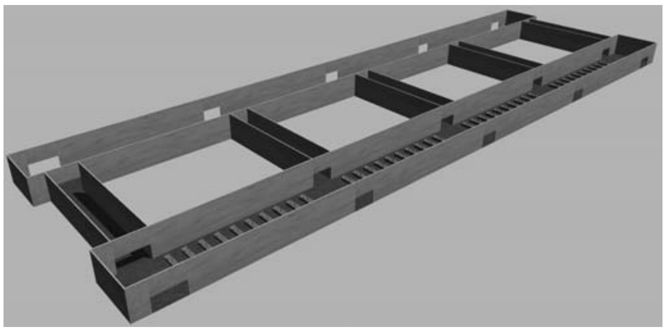


Figure 7.13: View of the model of two potrooms without markers

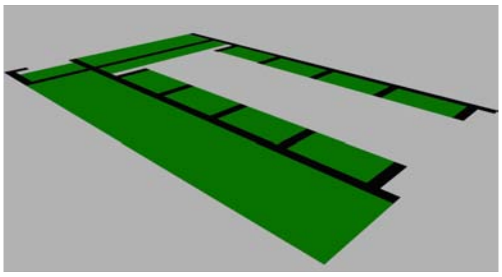
AFSH

The Aluminium Fluoride Storage and Handling is a rectangular building, smaller than a Potroom, in which the stocks of aluminum fluoride, useful for the production of aluminum, are stored. Such room presents a unique door that allows the entry and exit of vehicles. The design of such structure has been performed in a way that is similar to the one used in the second step of the Potroom model definition, neglecting the representation of not fundamental elements such as the roof and the aluminium fluoride's stocks.

7.2.2 Streets and external environment

Figure 7.14: Example of Aluminium Smelter.

To increase the reality of the simulation and, in particular, to obtain a more suitable outdoor environment, a 3D external environment is created. Drawing inspiration from some real aluminium smelters, (Figure 7.14) a simplified map is modeling using Blender. Firstly, in order to not overload the simulations, only the roads which link the AFSH to the Potroom and the "no-street areas" where the Fluoride Feeder can not pass through were defined; all the not influential objects such as other buildings and natural elements are not taken into account. An easy way to design the elements of the map is to define them as a concatenation of rectangular planes, black for the roads and green for the no-street areas. Finally, the Potroom and the AFSH buildings are added to obtain the complete simulation map where the Flouride Feeder can move in. Moreover to perform outdoor localization, Wi-Fi transceivers are placed in the external environment at 5 $[m]$ height. Wi-Fi antennas position with respect to the world frame are summarized in Table A.1, Table A.2, Table A.3 and Table A.4. Note that, due to the different outdoor simulations performed (Section 8.2), four wireless transceivers position layout needs to be defined. The 3D resulting simulation map is depicted in Figure 7.15 where the transmitters are not showing due to their small dimensions.


(a) 3D world map without Potroom and AFSH

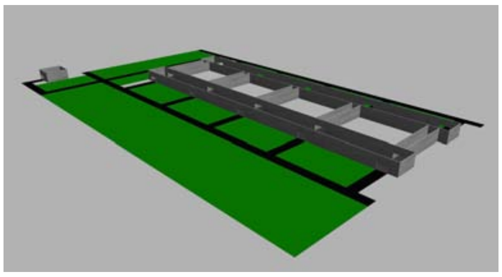

(b) 3D complete world map

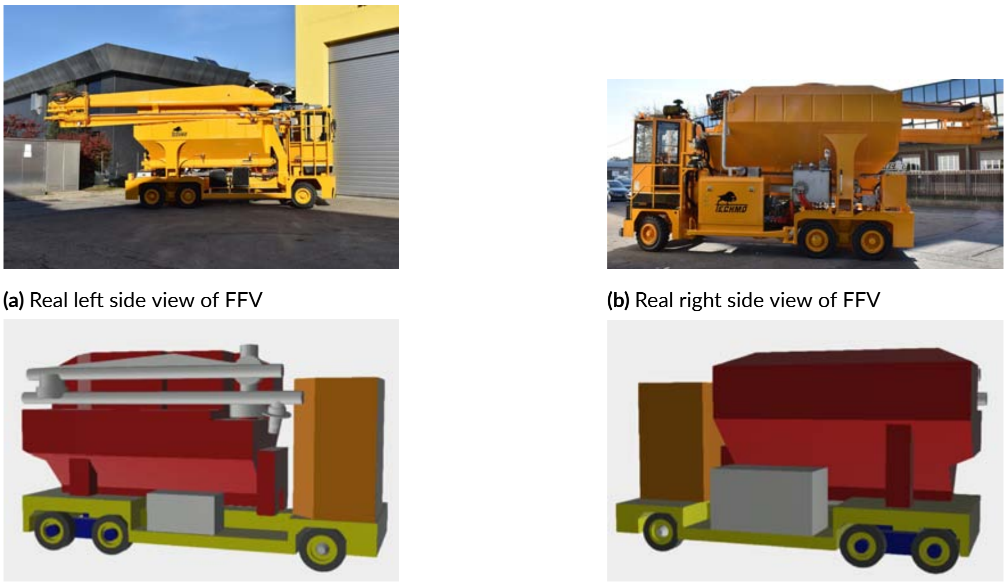
Figure 7.15: 3D world map for Gazebo simulation.

7.3 Fluoride Feeder modeling

To obtain a suitable fluoride feeder vehicle simulation model, it is helpful to create a model closest to reality in terms of dimensions, shape and description parameters of each vehicle components which can be affected the physic behaviour of the FFV itself. Indeed, the physics engine of Gazebo simulator is capable to well simulate gravity effect: thus, a correct mass and inertial matrix value of the vehicle is also fundamental for high accuracy simulation.

7.3.1 Vehicle Modeling

Firstly, started from the Autodesk Inventor 3D complex model of the FFV, each single element of the assembled model is extrapolated and exported in the 3D *.obj* file, useful for next steps; furthermore, the descriptive parameters like mass and inertial matrix are retrieved using the Autodesk Inventor *iProprieties* function (Table 7.2). The 3D model of each component is imported in the Blender software (Section 7.1.2) and, after simple color and texture elaboration, exported in the 3D *.dae* file. The latter is used as 3D mesh for the visual tag of the link elements of the fluoride feeder vehicle *.urdf* file. The result can be seen in Figure 7.16 and Figure 7.17 for each specific FFV parts. Now, the vehicle is ready to be displayed in Gazebo. To spawn the vehicle model and the simulated world (Section 7.2) in Gazebo, meaning to place them in the current position and orientation inside the Gazebo environment, a *.launch* file needs to be created. The latter is an XML format file used to define all the models, sensors, plugins and parameters required for the experiments. The *roslaunch* terminal command call the related *.launch* file previously created and spawning all the elements containing in the XML file. The transformation from code to simulated world is transparent to the user and, thus, not described in this thesis.


(a) Real left side view of FFV

(b) Real right side view of FFV

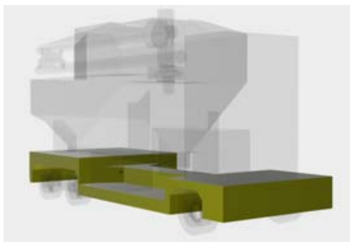
(c) Virtual left side view of FFV

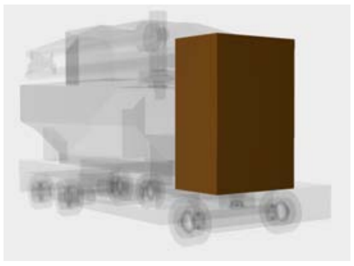
(d) Virtual right side view of FFV

Figure 7.16: Top view of feeding operation

Component name	Mass	X	Y	Z	I_{xx}	I_{yy}	I_{zz}
Frame	3143.09	2.20	6.10	0.72	11307.59	1522.91	12545.74
Cabin	662.98	1.40	1.30	2.24	509.87	465.22	329.16
Empty Container	3685.47	2.20	5.03	2.83	10410.34	409.83	9449.36
Full Container	14959.17	2.20	5.03	2.83	25648.30	9829.79	23811.73
Engine Group	1070.00	0.86	1.80	1.08	445.26	173.16	374.92
Oil Tank	425.80	0.53	1.19	0.73	86.00	38.39	84.16
Traction Axle	429.78	0.75	0.36	1.50	17.74	87.94	79.79
Steering Axle	285.69	1.55	0.39	0.15	2.50	1.98	42.73
Wheel	91.05	0.72	0.23	0.72	3.38	5.98	3.38
Hub	63.00	0.34	0.32	0.32	0.34	0.87	0.86

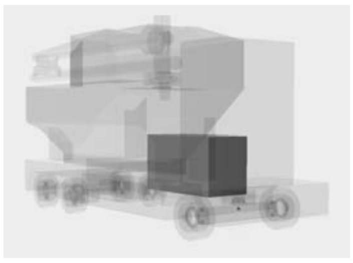
Table 7.2: Descriptive parameters of FFV components: dimensions [m], weigths [kg] and intertial moments [kgm^2]

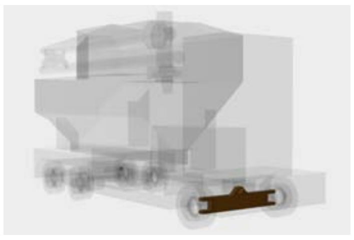

(a) Frame of FFV

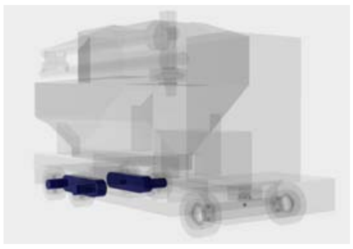

(b) Cabin of FFV

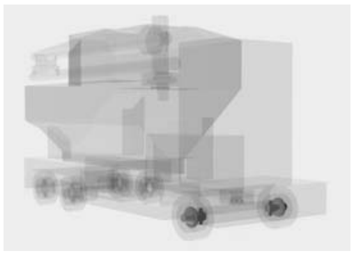
(c) Container of FFV

(d) Oil tank of FFV


(e) Engine group of FFV


(f) Steering axle of FFV


(g) Traction axle of FFV


(h) Hubs of FFV

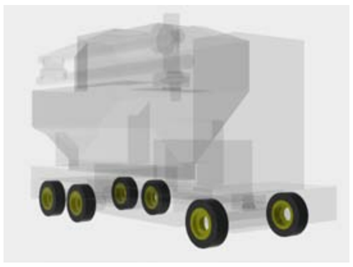

(i) Wheels of FFV

Figure 7.17: Virtual elements of the fluoride feeder vehicle

7.3.2 Fluoride Feeder Vehicle Sensors Modeling

As described in Section 7.1.5, vehicle can interact with the surrounding environment using sensors. The chosen sensors for the FFV model vehicle are:

- Cameras
- Wi-Fi receiver
- Laser scanners
- Inertial Measurement Unit (IMU)

To achieve real-time simulation speed, all the sensor are designed like simple withe boxes: no computational heavy meshes are attached to the sensors link.

Cameras

Three visual cameras are mounted over the cabin of the fluoride feeder vehicle in pre defined position and orientation with respect to the FFV coordinate frame (Table 7.4). They are used for the indoor fiducial markers system localization: they acquire frames with a certain resolution and frame rate (Table 7.3) and send the obtained images to the localization system for the marker detection process (Section 4.1.1 and Section 4.1.2). To better simulate a real camera, an *Additive White Gaussian Noise* (AWGN) $\mathcal{N}(0, 0.007)$ is added to camera pixels; note that this noise is sampled independently per pixel of each frame.

Camera Specifications	
Frame and Topic Rate	15 [*Frame/s*]
Frame Resolution	800x800 [*pixels*]
Image Color Space	8-bit RGB
Horizontal Field Of View	1.3962634 [*rad*]
Noise (AWGN)	$\mathcal{N}(0, 0.007)$

Table 7.3: Camera specifications

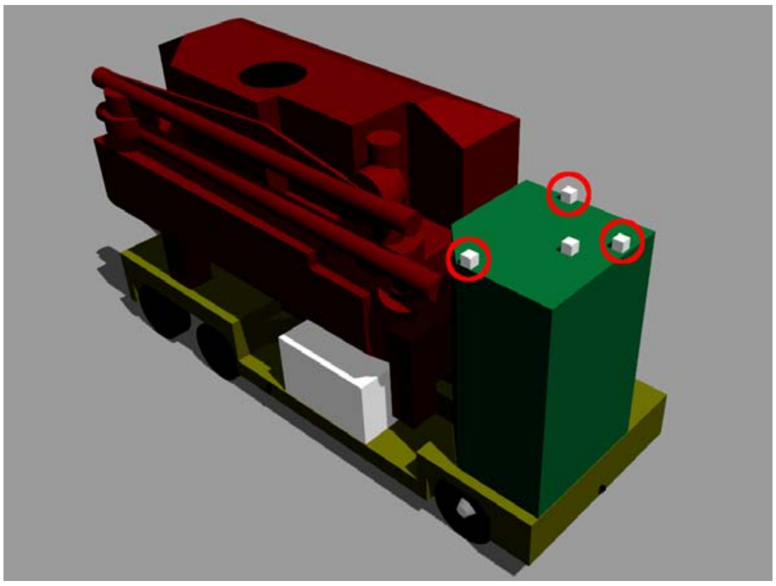


Figure 7.18: Camera pose on the fluoride feeder vehicle

Camera name	$\mathbf{R}_{\text{camera}}^{\text{vehicle}}$	$\mathbf{T}_{\text{camera}}^{\text{vehicle}}$
Right Camera	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.95 & 2.8 \end{bmatrix}^T$
Front Camera	$\begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.75 & 0 & 2.8 \end{bmatrix}^T$
Left Camera	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.35 & 2.8 \end{bmatrix}^T$

Table 7.4: Camera pose w.r.t. the FFV coordinate frame. The values of the translation vector have metric measure unit

Wi-Fi Receiver

To perform outdoor localization, the fluoride feeder vehicle needs to be equipped with a Wi-Fi receiver used to retrieve the *RSSI* value by each Wi-Fi transceivers placed in the external environment. As depicted in Figure 7.19, one Wi-Fi receiver is placed over the FFV cabin, in order to obtain the cleanest possible received signal strength indicator value avoiding object occlusion caused by the FFV components (container, retractile arm, etc.). Due to the huge outdoor dimensions, each transmitter has an high signal power, meaning that its wireless signal can be intercepted by the receiver even at long distances. Moreover, to simplified the simulation, Wi-Fi transmitters and receiver have same specifications (Table 7.5).

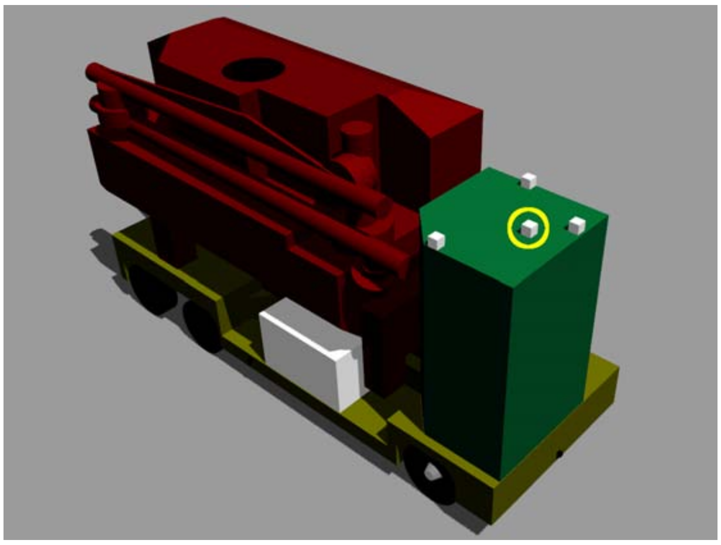


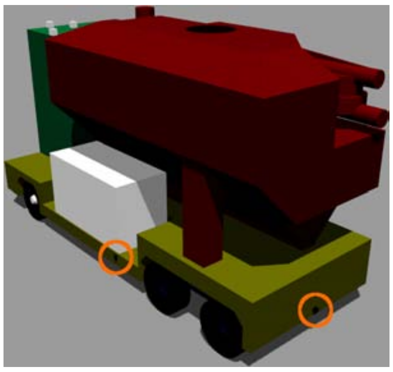
Figure 7.19: Wi-Fi receiver on the fluoride feeder vehicle

Wi-Fi Receiver and Transmitter Specifications	
Signal Rate	15 $[Hz]$
Frequency	2442 $[MHz]$
Gain	2.5 $[dB]$
Power	50 $[dBm]$ = 100 $[W]$

Table 7.5: Wi-Fi transmitter and receiver specifications

Laser Scanners

An automated guided vehicle performs different tasks in different environment where fixed or movable objects are already present. Moreover, most of the times, AGVs work in promiscuous area where they have to cooperate with other vehicle and workers. Thus, the fluoride feeder vehicle is equipped with multiple 2D laser scanners for safety reasons: with the laser scanners, the FFV is capable to detect surrounding obstacles and stop itself if the detected object is too close. Lasers are placed in order to obtain a full coverage of the FFV surrounding area to increase its adaptability to the environment changes. On the FFV, four 2D laser scanners are mounted, one on each side of the vehicle (Figure 7.20). Note that, because of the small obstacle which can interfere with the FFV tasks, the lasers are placed close to the bottom of the vehicle's frame. Finally, to make the laser scanner behaviour more realistic and similar to a real one, an AWGN noise is added.


(a) Back and left laser scanners

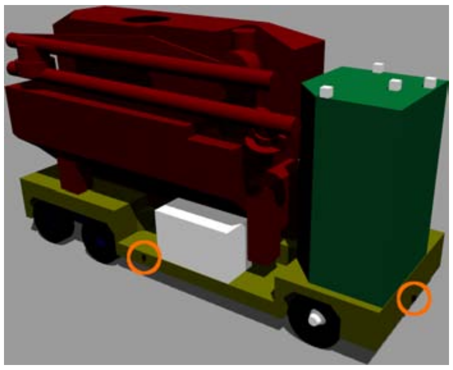

(b) Front and right laser scanners

Figure 7.20: Laser scanners on fluoride feeder vehicle

2D Laser Scanner Specifications	
Topic Rate	15 [*Hz*]
Angle Scan	180 [*deg*] with resolution of 0.25 [*deg*]
Scan Range	0.1 - 50.0 [*m*]
Noise	$\mathcal{N}(0.0, 0.01)$

Table 7.6: 2D laser scanner specifications

Inertial Measurement Unit

The Inertial Measurement Unit sensor (Section 6.4.1) is used to retrieve information about orientation ([*rad*]), angular speed ([*rad/s*]) and linear acceleration ($[m/s^2]$) with respect to the FFV reference frame. The accelerometer measures the linear acceleration of the FFV considering the external forces with respect to the FFV reference frame, that is

$$\mathbf{a}_{IMU} = \mathbf{R}^T(\alpha, \beta, \gamma)(\ddot{\mathbf{p}} - g\mathbf{e}_3) + \mathbf{b}_{\mathbf{a}} + \boldsymbol{\eta}_{\mathbf{a}} \tag{7.2}$$

The gyroscopes measures the angular velocity of the FFV with respect its reference frame, so that

$$\boldsymbol{\omega}_{IMU} = \boldsymbol{\omega} + \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega} \tag{7.3}$$

where for both accelerometer and gyroscope $\mathbf{b}$ and $\boldsymbol{\eta}$ are, respectively, the bias error and the measurement noise. Note that the bias error can be compensated via sensor calibration.
Its information are merged with the odometry ones using the Extended Kalman filter (Section 6.4) to improve localization precision and accuracy. The IMU sensor (no image because the sensor is not visible) is mounted on the FFV in the origin of the fluoride feeder vehicle 3D coordinate system and it send data with a frequency of 15 [*Hz*]. To make the sensor and all the simulation more realistic, an additive Gaussian white noise is added to the IMU measurements, that is $w \sim \mathcal{N}(0.0, 0.1)$. Finally, as already mentioned in Section 6.4.1, the sensor used in the simulation is an ideal inertial measurement unit, meaning with high orientation accuracy, no accelerometer bias and with small gyroscope measurement error.

8
Experimental Results

This chapter describes and presents the experimental results obtained performing simulations for the indoor and outdoor FFV localization using the Gazebo Simulator software and the ROS framework (Section 7.1.5 and Section 7.1.4). In particular, for each localization technique, two types of simulations have been performed:

1. **Stationary vehicle localization**: the FFV is placed in the simulation environment with fixed position and orientation; the simulation runs for about 15 $[s]$ and only the odometry information are used to localize the FFV. Then, information retrieved using fiducial markers or Wi-Fi techniques are combined with the inertial measurement unit data using the EKF in order to obtained better performances.

2. **Moving vehicle localization**: the FFV is localize while is moving from a starting position to an ending one defined before the simulation starts. To test the localization robustness and reliability while the FFV turns, a non linear trajectory is performed by the vehicle. As already described in the stationary simulations, odometry data and IMU information are merged together using the EKF.

For the indoor localization, to deeply analyze the fiducial markers technique, both position and orientation data obtained by the tags detection process are used in the EKF. Hence, orientation information retrieved by the inertial measurement unit sensor are neglected. For the outdoor localization, the relation between the number of wireless transmitters and localization accuracy is also tested. Indeed, localization process are performed, on stationary FFV, using 4,8 and 14 transmitters each time. The latters are placed in the outdoor environment asymmetrically to not distort the estimated position error with respect to the true one. Finally, only the 2D position coordinates *x* and *y* and the *yaw* orientation angle are acquired and considered as relevant for the thesis goal. Indeed, a planar simulation environment is considered.

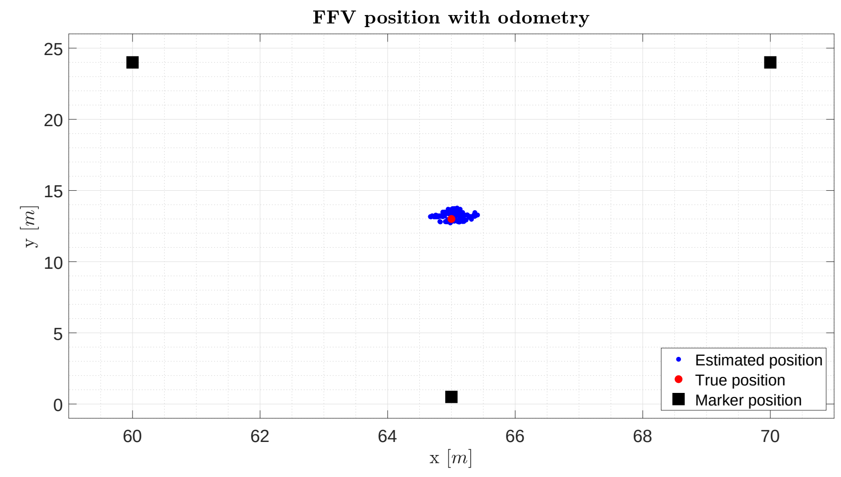
8.1 Indoor localization experimental results

8.1.1 Stationary vehicle localization

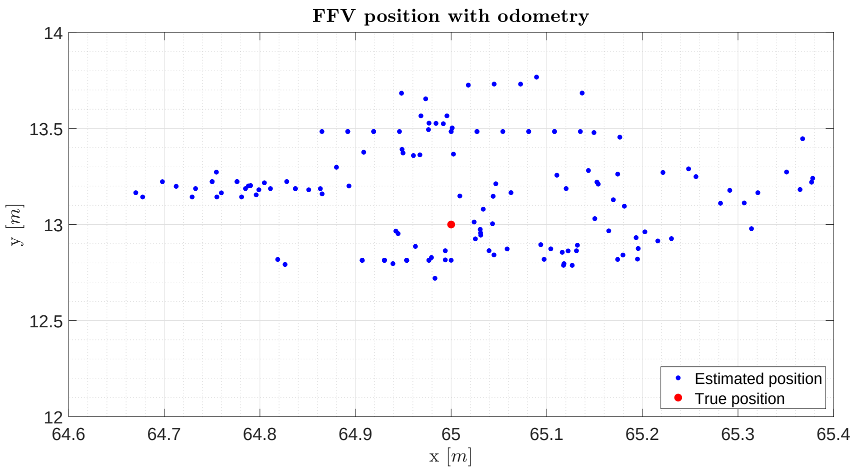
For this simulation, the FFV is placed inside the potroom with fixed position and orientation, that are

$$\begin{cases} \mathbf{x}[m] = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} 65 & 13 & 0.35 \end{bmatrix}^T \\ \boldsymbol{\theta}[rad] = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \end{cases} \tag{8.1}$$

The simulation is performed for about 15 [*s*] which takes into account the EKF *initialization time* which is about $2-3$ [*s*]. The latter is fundamental to stabilize the filter around the initial estimated position and orientation of the FFV, retrieved with the markers detection. Drawing inspiration from Figure 8.1a, the vehicle position retrieved by detecting only the markers oscillates, along the *x*-axis, from 64.67 to 65.39 [*m*], while, along the *y*-axis, from 12.72 to 13.77 [*m*]. Moreover, only three tags are detected to perform this stationary localization: the first (on the bottom), which is perpendicular to the right camera and the remaining two (the top ones) which have both a planar orientation with respect to the optical axis of the left camera of 0.43 [*rad*] ($24.5°$). Moreover, the top right marker has, also, planar orientation with respect to the optical axis of the front camera of 1.15 [*rad*] ($65.90°$).



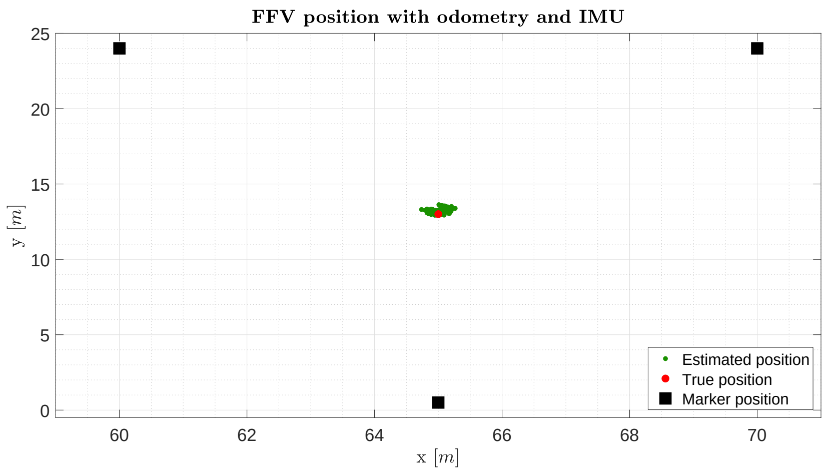
(a) Stationary FFV position with odometry



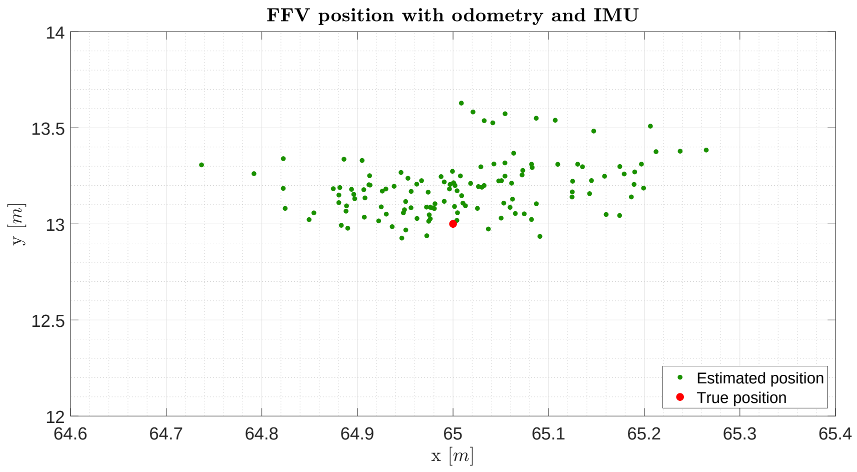
(b) Stationary FFV position with odometry (zoom on 64.6 - 65.4 [m] along the x axis)

Figure 8.1: Stationary FFV indoor position with odometry

As depicted in Figure 8.1b, the estimated position (blue dots) is shifted over red dot, from the vehicle and the non perpendicular tags. Moreover, the top blue dots (estimated position obtained using the non orthogonal markers) are more scattered than the bottom ones (estimated position obtained using the perpendicular tag). Concerning this raw stationary simulation, markers orientation with respect to the cameras optical axis is a key parameter for the localization performance. Fig 8.2 depicts the FFV stationary localization performed using the sensor fusion technique with the EKF. The simulation setup is the same used for the previous one but, in this case, data acquisition started after the initialization time of the filter. As can be seen from Figure 8.2b, the accuracy, in terms of localization range on the x and y axis, has increased: vehicle position goes, along the *x*-axis, from 64.74 to 65.26 $[m]$, while, along the *y*-axis, from 12.93 to 13.63 $[m]$.



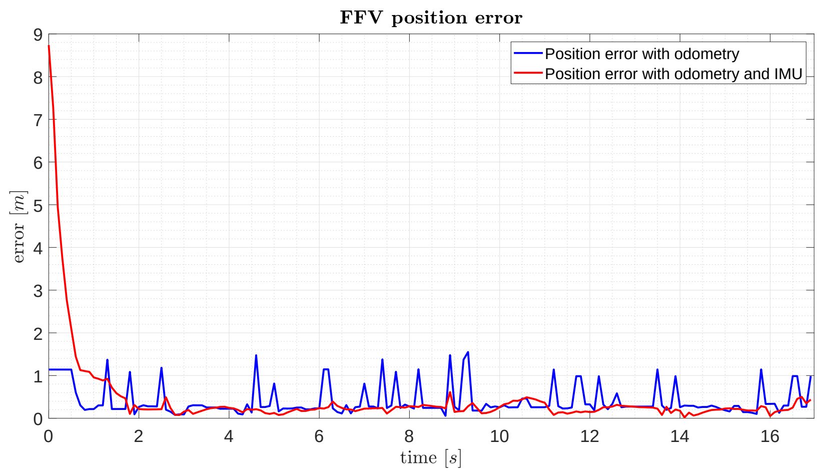
(a) Stationary FFV position with odometry and IMU



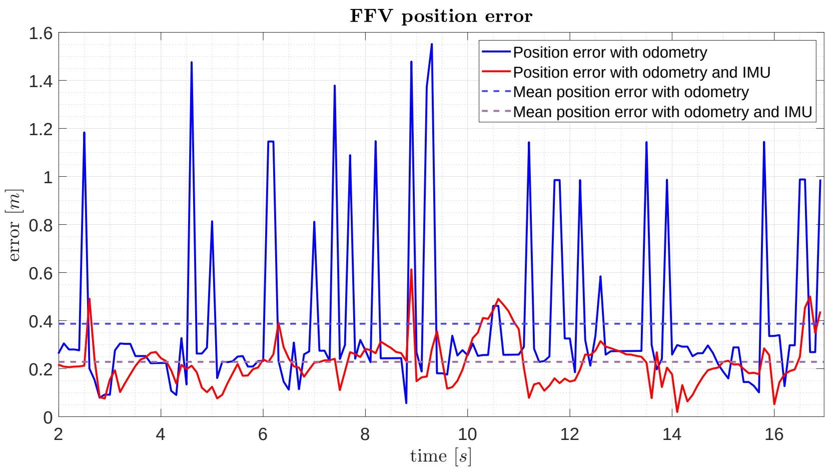
(b) Stationary FFV position with odometry and IMU (zoom on 64.6 - 65.4 [m] along the x axis)

Figure 8.2: Stationary FFV indoor position with odometry and IMU

Note that, because of the pose information used on the EKF are obtained from the markers detection process, the estimated vehicle position computed with the sensor fusion approach is affected by the poor accuracy localization performed by the left and front cameras with the non perpendicular tags. Indeed, observing the Figure 8.2b, the green dots matching the FFV position is shifted to the top with respect to the true vehicle position.



(a) Stationary FFV position error with odometry and odometry + IMU



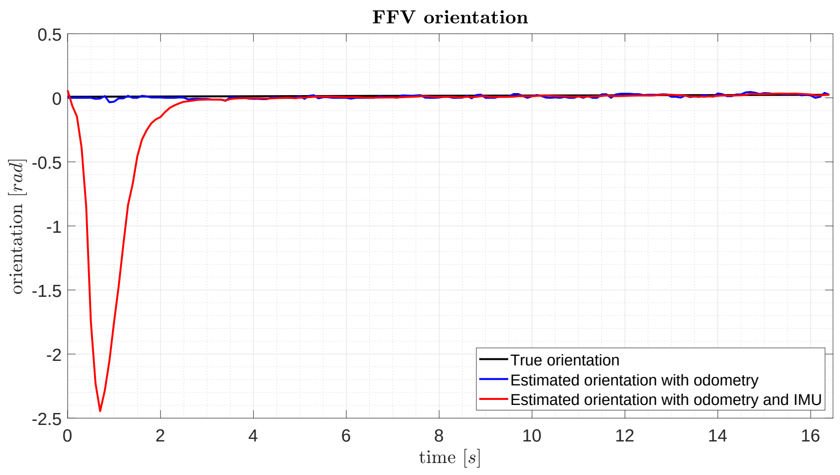
(b) Stationary FFV position error with odometry and odometry + IMU zoomed on 2 - 17 [s] along the x axis

Figure 8.3: Stationary FFV indoor position error with odometry and odometry + IMU

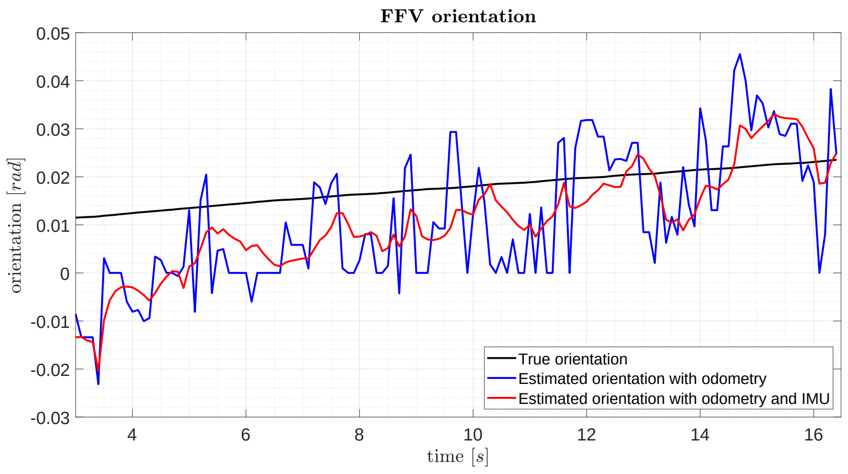
Indoor stationary position error results are summarized in Figure 8.3 and Table 8.1: in particular, in Figure 8.3a, it's possible to notice the position error computed with the sensor fusion technique tends to its mean value after the initialization time of the filter. In conclusion, sensor fusion technique, applied to the stationary localization, increase the estimated position accuracy of about 41% with respect to the one computed using only the fiducial markers localization approach.

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	0.3869	0.1170	0.3421
Odometry and IMU	0.2282	0.0096	0.0980

Table 8.1: Mean, variance and std. dev. of the FFV position error



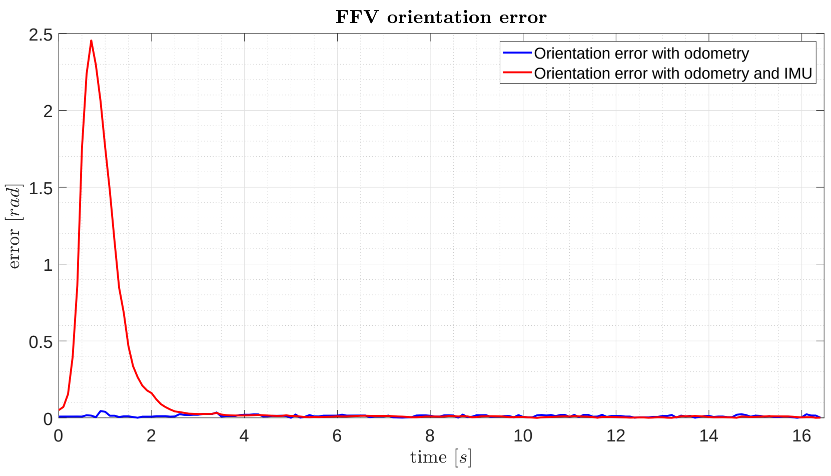
(a) Stationary FFV orientation with odometry and odometry + IMU



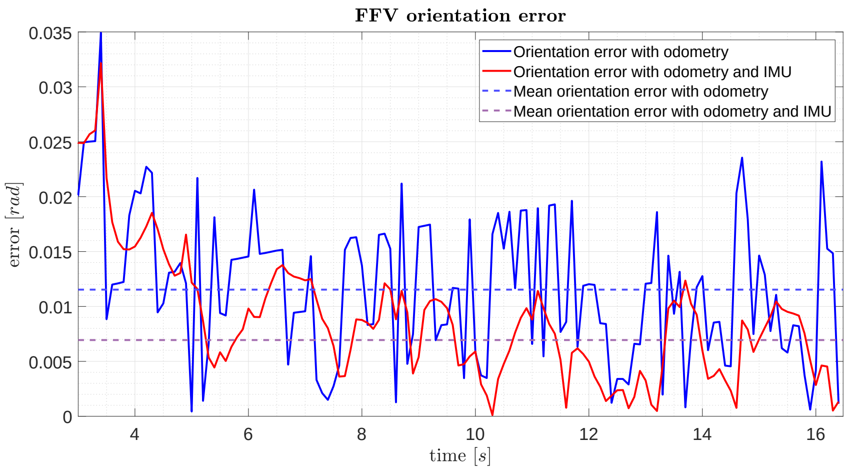
(b) Stationary FFV orientation with odometry and odometry + IMU zoomed on $3 - 16$ [s] along the x axis

Figure 8.4: Stationary FFV indoor orientation with odometry and IMU

Drawing inspiration from Figure 8.4, estimated orientation is almost similar to the true one, using both the localization techniques. In particular, the sensor fusion technique gives smoother results with respect to the markers detection one (Figure 8.4b). Note that, observing the black line relative to the true orientation, even if the vehicle is stationary, an orientation draft phenomenon takes place (the FFV turns even if it is fixed in a predefined position).



(a) Stationary FFV orientation error with odometry and odometry + IMU



(b) Stationary FFV orientation error with odometry and odometry + IMU zoomed on 3 - 16 [s] along the x axis

Figure 8.5: Stationary FFV indoor orientation with odometry and IMU

Orientation error results are depicted in Figure 8.5 and Table 8.2: sensor fusion technique increase the estimated orientation accuracy of about 40% with respect to the one estimated using only the markers detection process.

	Mean Error [m]	***Variance [m]***	***Std. Dev. [m]***
Odometry	0.0115	5.12×10^{-5}	0.0072
Odometry and IMU	0.0069	1.09×10^{-5}	0.0033

Table 8.2: Mean, variance and std. dev. of the FFV orientation error

In conclusion, stationary indoor localization gives good results for both estimated position and orientation. Moreover, sensor fusion technique increases the localization accuracy, meaning the mean error computed between estimated pose and the true one is lower than the one calculated using only the markers localization process. However, markers orientation with respect of the cameras optical axis affects the localization performances resulting in more oscillating estimated pose values. To attenuate this phenomenon and to increase the estimated pose accuracy, a suitable threshold of the planar angle between camera and tags has to be chosen (in this thesis the threshold is set to 1 [*rad*]).

8.1.2 Moving vehicle localization

For this second indoor localization simulation, the FFV needs to follow an *"S"* trajectory which leads the FFV from the first potroom to the second one. The path presents two curves, one, to the left, for which the FFV goes from the potroom to the corridor and the second one, to the right, for which the vehicle goes from the hallway to the second potroom. Starting and ending points of the trajectory, initial and final vehicle orientation, linear velocity and initial orientation are defined before the simulation starts, that are:

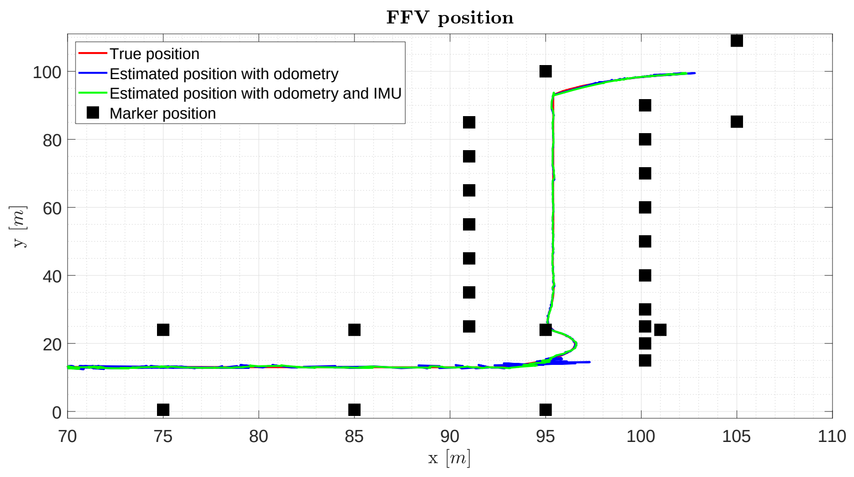
$$\begin{cases} \mathbf{x_0}[m] = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} 70 & 13 & 0.35 \end{bmatrix}^T \\ \mathbf{x_1}[m] = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} 103 & 100 & 0.35 \end{bmatrix}^T \\ \boldsymbol{\theta_0}[rad] = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{\theta_1}[rad] = \theta_\mathbf{0} \\ \mathbf{v}[m/s] = 5 \end{cases} \tag{8.2}$$

Note that, from Equation 8.2, the linear velocity is chosen arbitrarily under the 10 [*km/h*]: the latter is almost the real maximum velocity achieved by the FFV inside the potroom. As already seen in the previous section, both the only marker detection and the sensor fusion localization are tested and compared with the true pose of the FFV; information about mean, variance and standard deviation error of the vehicle estimated pose are depicted in the next tables and graphs.

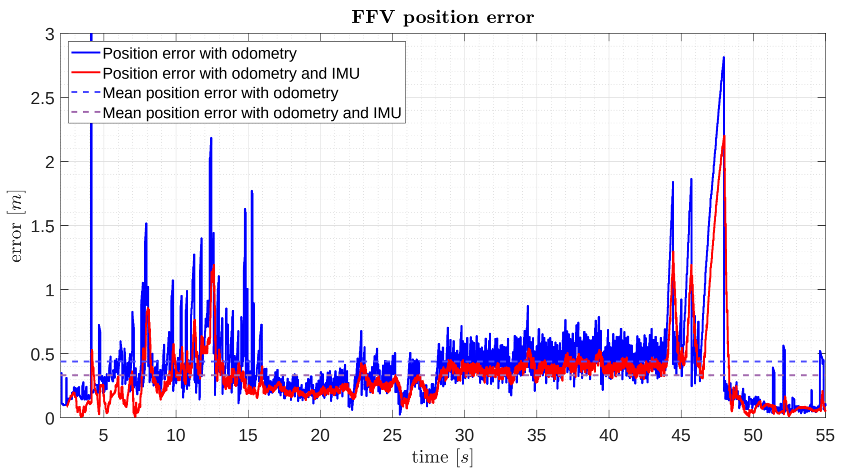
Drawing inspiration from Figure 8.6, both localization techniques have good accuracy performances during almost the entire path. In particular the sensor fusion localization is more accurate than the other one, even when the FFV turns to enter and to get out the corridor, which causes oscillations and errors in the marker detection process. These phenomena are caused by the fast and sudden direction change computed by the vehicle when it starts turning. Moreover, in the curves, tags are not perpendicular to the optical axis of the cameras, causing both misleading markers detection and low accurate estimated pose.

Observing the position error plot in Figure 8.6b, it's possible to split data into three separate areas. The first one, from 0 to 12.5 [*s*] concerns the first part of the path where markers on the walls are far away from the vehicle (at least 12[*m*]) and where the vehicle has to turn left to enter in the corridor. The second one, from 12 to 36 [s] concerns the second part of the path where the vehicle is crossing the corridor. In this sector of the building, markers are close to the vehicle (at most 3 [*m*]) due to the narrow width between walls. Moreover, the FFV performs a linear trajectory. The last one, from 36 to 45 [*s*] concerns the third part of the path which includes the right turn: the latter is performed with high speed in order to test the marker detection in presence of fast direction change.

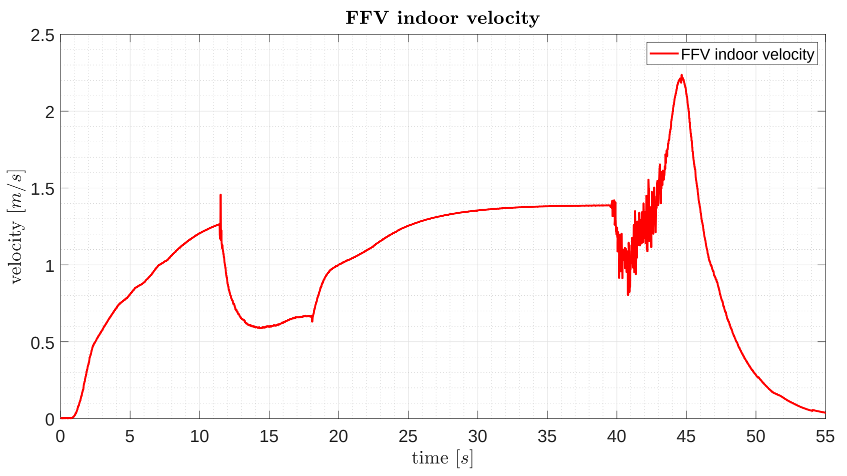
Markers localization technique is very effective for linear trajectory with tags near to the vehicle, while it lose accuracy when tags distance, with respect to the vehicle, is over 10 [*m*]. Moreover, during the left and right turns, position accuracy is affected by the angular velocity achieved by the vehicle to compute the turn and the radius of the curve itself: "soft" turn like the one performed to enter in the corridor with low speed and big turn radius gives better localization performances than the "hard" one computed to exit from the corridor with high speed and small turn radius. Position error results are depicted in Figure 8.6b and Table 8.3:



(a) Moving FFV indoor position with odometry and odometry + IMU



(b) Moving FFV indoor position error with odometry and odometry + IMU



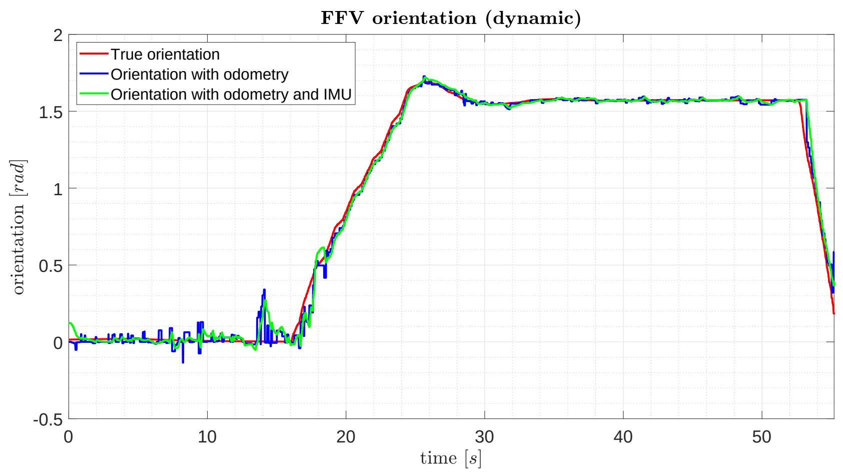
(c) Moving FFV indoor velocity

Figure 8.6: Moving FFV indoor position: starting point is set to $(x, y) = (70, 13)[m]$

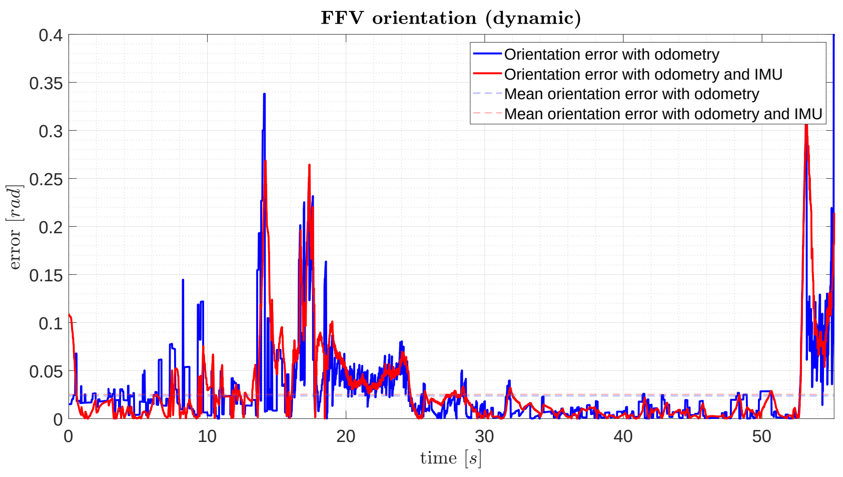
sensor fusion technique increase the estimated position accuracy of about 19% with respect to the one estimated using only the markers detection approach.

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	0.3748	0.1628	0.4035
Odometry and IMU	0.3046	0.0744	0.2728

Table 8.3: Mean, variance and std. dev. of the FFV position error



(a) Moving FFV indoor orientation with odometry and odometry + IMU



(b) Moving FFV indoor orientation error with odometry and odometry + IMU

Figure 8.7: Moving FFV indoor orientation error with odometry and odometry + IMU

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	0.0237	0.0013	0.0366
Odometry and IMU	0.0254	0.0014	0.0369

Table 8.4: Mean, variance and std. dev. of the FFV orientation error

For the orientation information retrieved using both the localization methods (Figure 8.7), same considerations and conclusion just outlined for the position data can be defined. Indeed, close tags distance from the vehicle and linear path trajectory give the best results of the entire simulations, while distant tags and direction changes affect significantly the orientation accuracy. In particular, let us consider the orientation data retrieved while vehicle performed the "hard" right turn. The estimated orientation, using both the localization approaches, does not match at all the true orientation values while the vehicle starts turning to enter in the second potroom (time interval $[52\ 52.5]$ $[s]$ of the simulation). The marker detection process is computationally too heavy to follow the fast vehicle dynamics in this small amount of time causing, combined with the fast direction change, a large orientation error. Orientation error results are depicted in Figure 8.7b and Table 8.4. Looking at the Table 8.4, it's interesting to observe how fast direction changes and vehicle dynamics can affected the orientation error using the sensor fusion approach. The latter is slightly worse than the one computed using only the markers detection technique. Indeed, the EKF, in order to work properly, needs to be implemented assuming that the dynamic of the vehicle is slower than the data acquisition and elaboration rate.

In conclusion, moving indoor localization gives good results for both estimated position and orientation, especially concerning the sensor fusion approach. As already discussed, tags distance and orientation with respect to the cameras central axis, angular velocity achieved by the vehicle during the turns and radius of the curves are crucial parameters to take into account to obtain a suitable localization. Indeed, estimate pose error can be reduce setting specific thresholds like maximum orientation angle and distance between tags and cameras (for this simulation the thresholds are set, respectively, to 1 $[rad]$ and 20 $[m]$). Moreover, if the vehicle dynamic is too fast for the EKF, it's possible to tune the data acquisition rate paying attention to not obtain too much time distant odometry and inertial information which can compromise the localization performances.

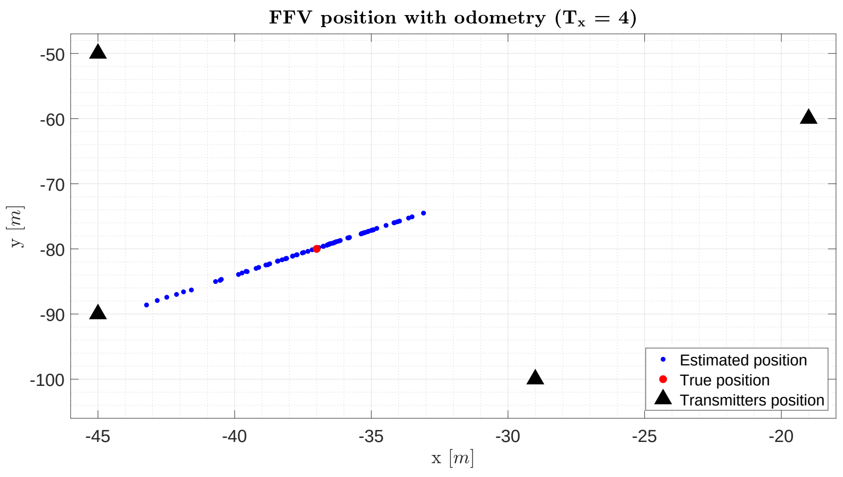
8.2 Outdoor localization experimental results

8.2.1 Stationary vehicle localization

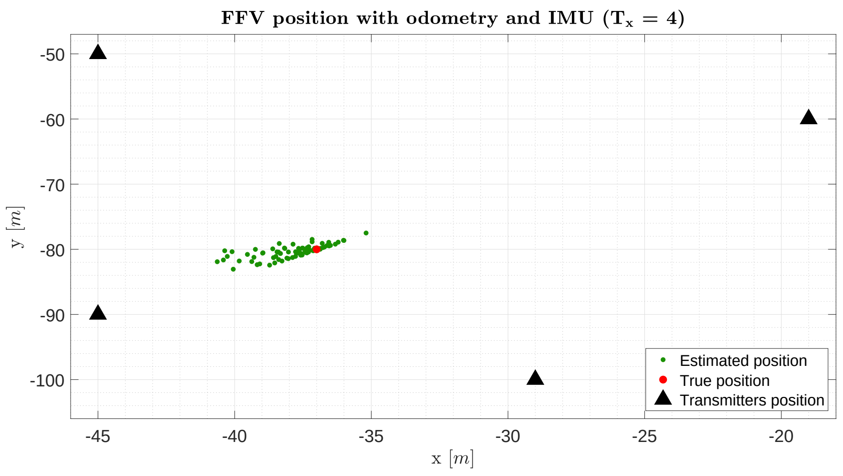
The first outdoor localization simulation is performed placing the FFV in a pre defined fixed position and orientation defined as follows:

$$\begin{cases} \mathbf{x}[m] = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} -37 & -80 & 0.35 \end{bmatrix}^T \\ \boldsymbol{\theta}[rad] = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 1.57 \end{bmatrix}^T \end{cases} \tag{8.3}$$

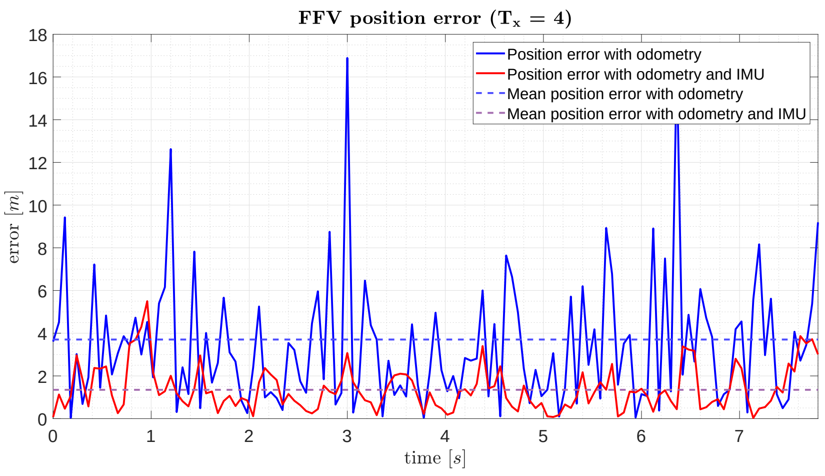
For localization purposes, only the 2D planar coordinates *x* and *y* and the *Y* angle are used. The simulation is performed for about 10 [*s*] including the EKF initializing time. Moreover, in order to analyze the relation between number of transmitters and localization performances, simulation is computed surrounding the vehicle with different number of Wi-Fi transmitters (4,8 and 14). Each Wi-Fi beacon is placed at 5 [*m*] height and located in the environment in order to avoid symmetries with respect to the vehicle pose. Indeed, symmetries can positively affected the vehicle position, distorting the real accuracy performances of the RSSI based Wi-Fi localization: for example, if the transmitters are placed with the same *y* axis, the estimated *y* coordinate of the vehicle position will be very close to the real one. Each simulation is computed bounding the possible RSSI acceptable values between 0 to -85 [*dBm*]. This outdoor localization is in free-space without any support or priori info by maps or other external supports.



(a) Stationary FFV outdoor position with odometry

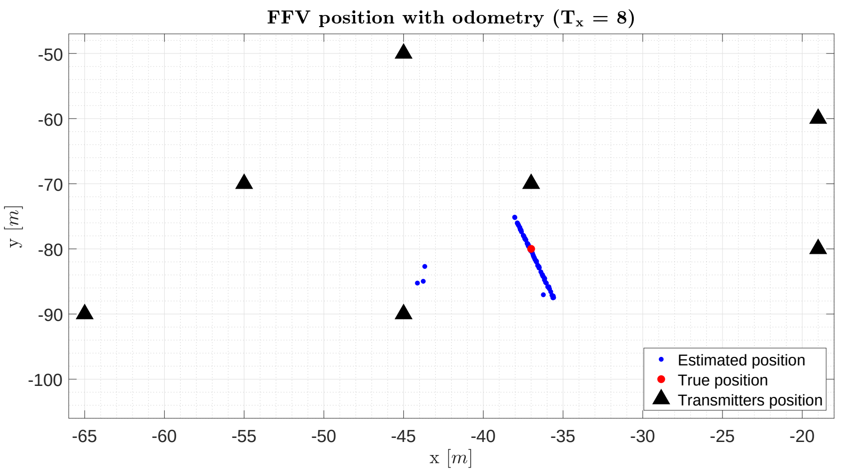


(b) Stationary FFV outdoor position with odometry and IMU

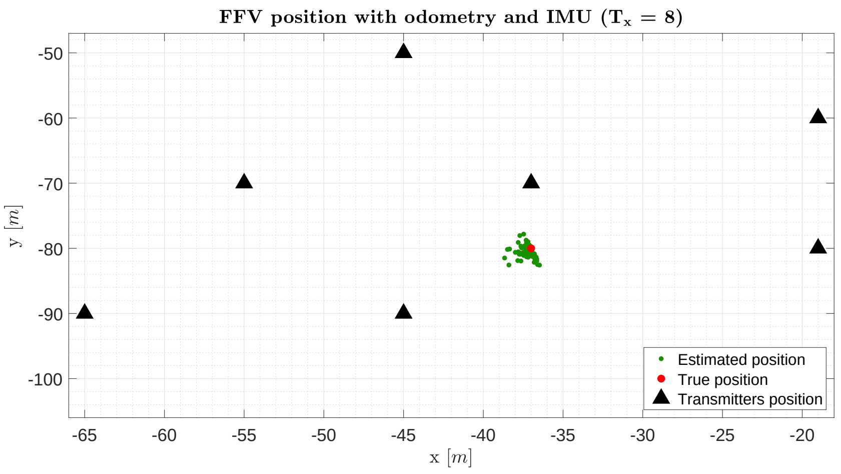


(c) Stationary FFV outdoor position error with odometry and odometry + IMU

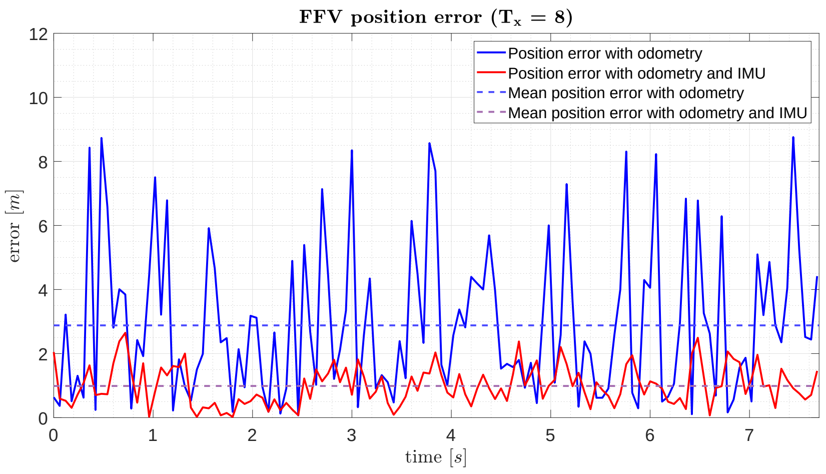
Figure 8.8: Stationary FFV outdoor position ($T_x = 4$)



(a) Stationary FFV outdoor position with odometry

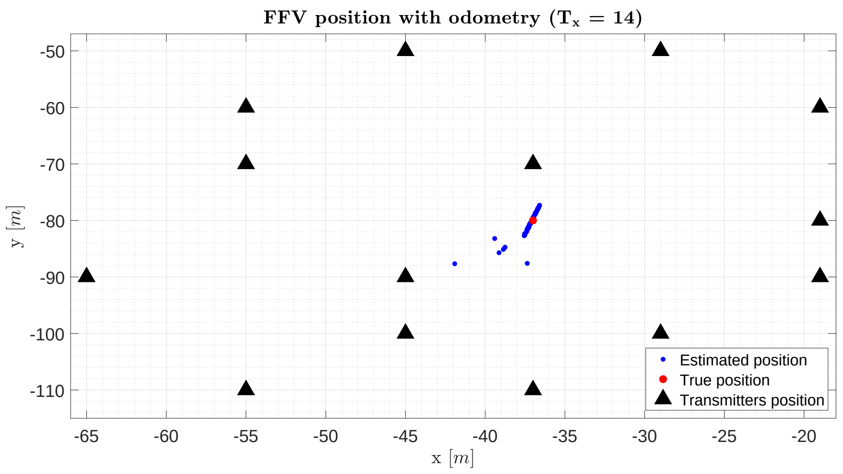


(b) Stationary FFV outdoor position with odometry and IMU

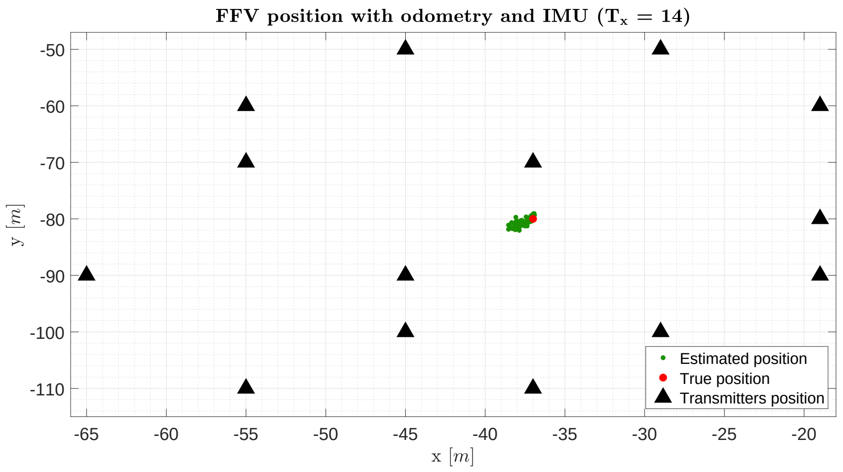


(c) Stationary FFV outdoor position error with odometry and odometry + IMU

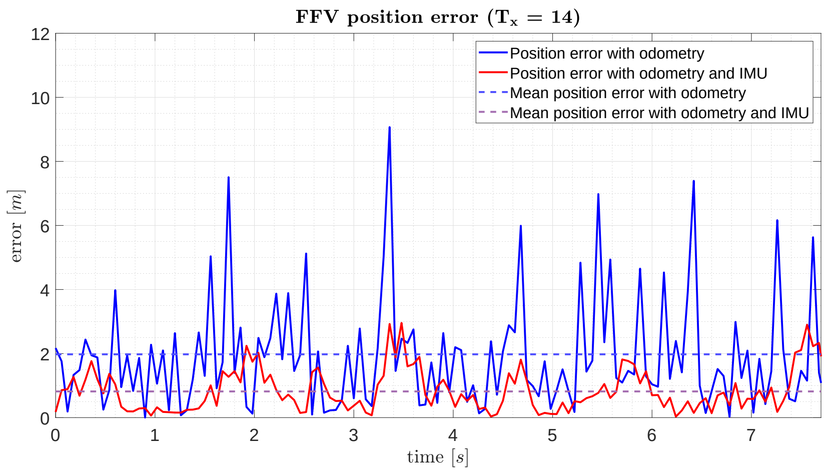
Figure 8.9: Stationary FFV outdoor position ($T_x = 8$)



(a) Stationary FFV outdoor position with odometry



(b) Stationary FFV outdoor position with odometry and IMU



(c) Stationary FFV outdoor position error with odometry and odometry + IMU

Figure 8.10: Stationary FFV outdoor position ($T_x = 14$)

Drawing inspiration from Figure 8.8, Figure 8.9 and Figure 8.10 the vehicle positions computed using the sensor fusion localization approach are significantly better than the ones computed using only the RSSI indicator approach. Moreover, the number of Wi-Fi transmitters employed in each simulation is directly correlated to the estimated position accuracy (Figure 8.11): increasing the number on antennas gives better position results reducing the mean error value. Position results for these stationary simulations are summarized in Table 8.5, Table 8.6 and Table 8.7.

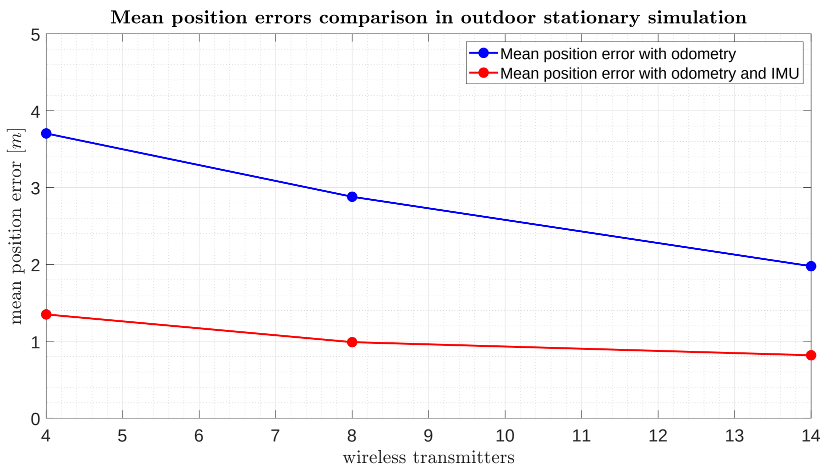


Figure 8.11: FFV outdoor mean error position comparison with different number of reachable wireless transmitters

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	3.7041	12.2972	3.5067
Odometry and IMU	1.3491	1.0305	1.0151

Table 8.5: Mean, variance and std. dev. of the FFV position error ($T_x = 4$)

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	2.8802	4.8916	2.2117
Odometry and IMU	0.9883	0.3822	0.6182

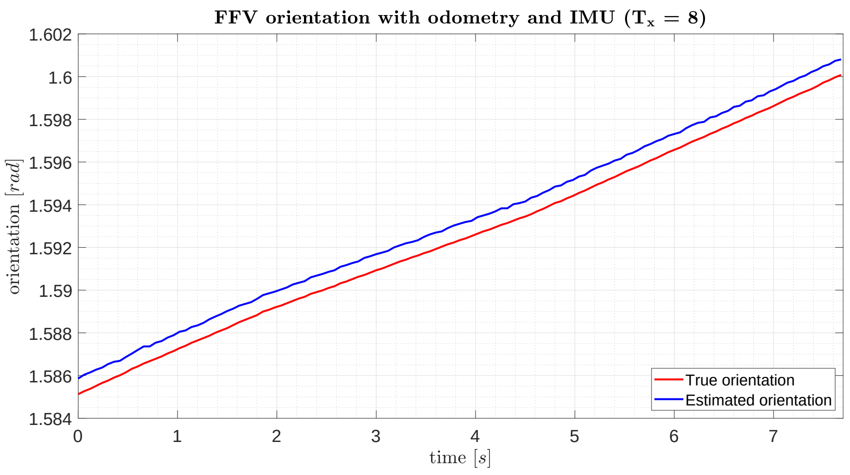
Table 8.6: Mean, variance and std. dev. of the FFV position error ($T_x = 8$)

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	1.9768	4.1929	2.0477
Odometry and IMU	0.8181	0.4093	0.6397

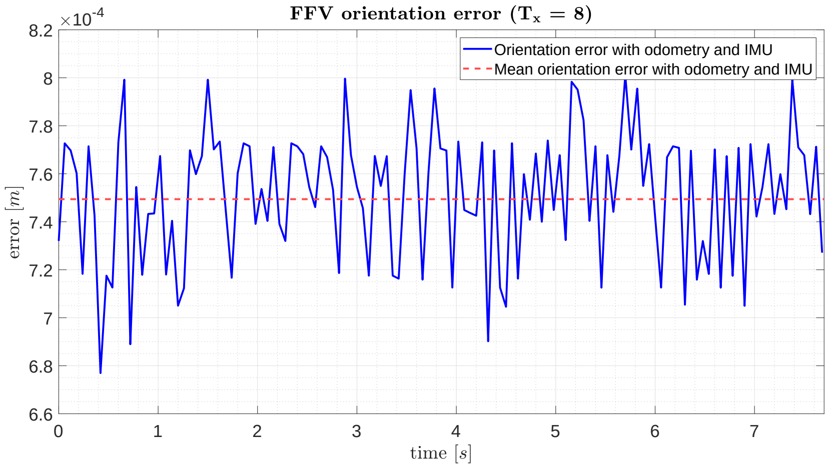
Table 8.7: Mean, variance and std. dev. of the FFV position error ($T_x = 14$)

Considering the position error tables with 4 and 14 transmitters, the mean position error using only the odometry information decreasing of about 47%, while the one computed using both odometry and inertial measurement unit information decreasing of about 39%. Moreover the estimated position obtained with the sensor fusion approach has better performance with the related only Wi-Fi based localization (almost 59% better). On the other hand, as depicted in Table 8.1, the position accuracy retrieved using the RSSI-based Wi-Fi technique for the stationary vehicle localization is significantly worse (about 70% position error more) than the one calculated in the indoor stationary simulations.

Observing the Figure 8.12, orientation error is close to zero, thanks to the estimated orientation which is similar to the true one. Firstly, note that in the outdoor localization, the orientation information are obtained using only the IMU sensor: this means that it's impossible to acquire orientation data using the only odometry information. Hence, only the sensor fusion approach is used for this simulation and, due to the fact that the orientation is computed independently to the presence and the number of the Wi-Fi transmitters, the estimated orientation values are almost equal using 4,8 or 14 antennas (for simplicity, only the graphs with $T_x = 8$ of the estimated orientation and its error is depicted in the thesis). However, as described in the section 6.4.1, the IMU sensor used for the simulation is ideal, meaning that the bias obtained with the accelerometer is almost zero and, moreover, the noise which affect the IMU gyroscope is very small. Orientation error results with 8 transmitters are depicted in Table 8.8.



(a) Stationary FFV outdoor orientation with odometry



(b) Stationary FFV outdoor orientation error with odometry and IMU

Figure 8.12: Stationary FFV outdoor orientation ($T_x = 8$)

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry and IMU	7.4941×10^{-4}	7.2484×10^{-10}	2.6923×10^{-5}

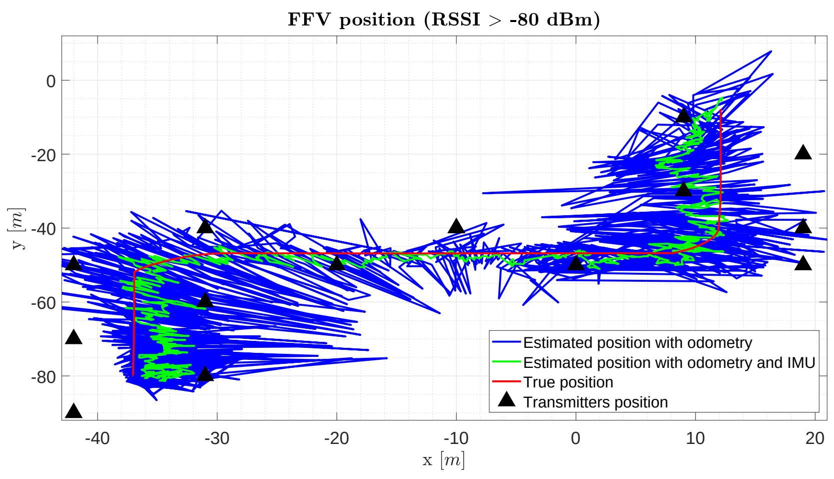
Table 8.8: Mean, variance and std. dev. of the FFV orientation error ($T_x = 8$)

In conclusion, the number of the wireless transceivers surrounding the stationary vehicle is directly correlated to the estimated position accuracy. Moreover, as already seen for the stationary indoor localization, sensor fusion localization approach is significantly better than the one performed using only the odometry information obtained with the RSSI-based Wi-Fi technique. On the other hand, increasing the number of Wi-Fi antennas results in more sophisticated localization infrastructure with higher installation and maintenance costs. Hence a cost benefits analysis needs to be done to determine the optimal number of wireless transmitters to obtain discrete localization performances.

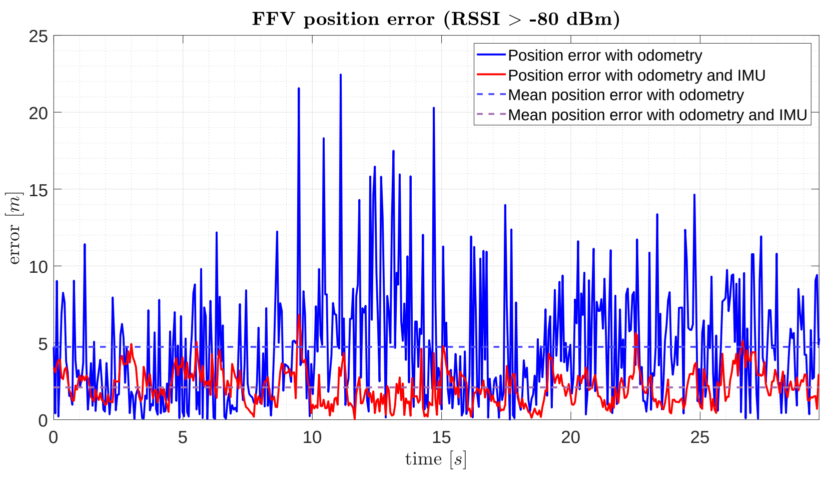
8.2.2 Moving vehicle localization

For the last experimental tests of the project, the vehicle, as already seen in Section 8.1.2, needs to follow a non linear path ("S" trajectory) staying over the streets defined in the external environment. The path presents two turns with same radius, the first to the left and the second to the right. Furthermore, the transmitters are placed along the roadsides at the height of 5 meters simulating a real street lamps network. To better understand how the wireless signal degradation over long distances can affect the localization performances, two simulations are defined using two different boundary on the RSSI indicator value. The first is $RSSI > -80$ [*dBm*], while the second is $RSSI > -75$ [*dBm*]. The latter is chosen more strictly than the other one to avoid that distant transmitters can erroneously affect the localization measure: roughly speaking, distant transmitters are considered as outliers. Initial and final position and orientation of the vehicle and its linear velocity are defined as follows:

$$\begin{cases} \mathbf{x_0} = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} -37 & -80 & 0.35 \end{bmatrix}^T \\ \mathbf{x_1} = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} 12.5 & -13 & 0.35 \end{bmatrix}^T \\ \theta_\mathbf{0} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 1.57 \end{bmatrix}^T = \theta_\mathbf{1} \\ \mathbf{v} = 5\ m/s \end{cases} \tag{8.4}$$



(a) Moving FFV outdoor position with odometry and odometry + IMU

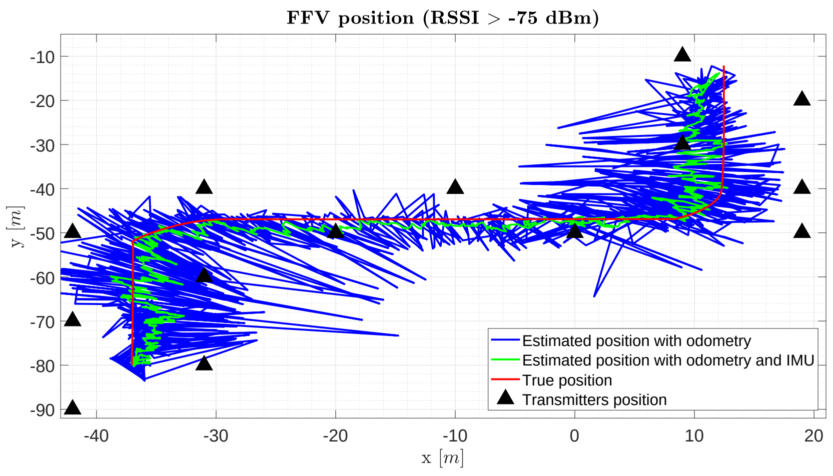


(b) Stationary FFV outdoor position error with odometry and odometry + IMU

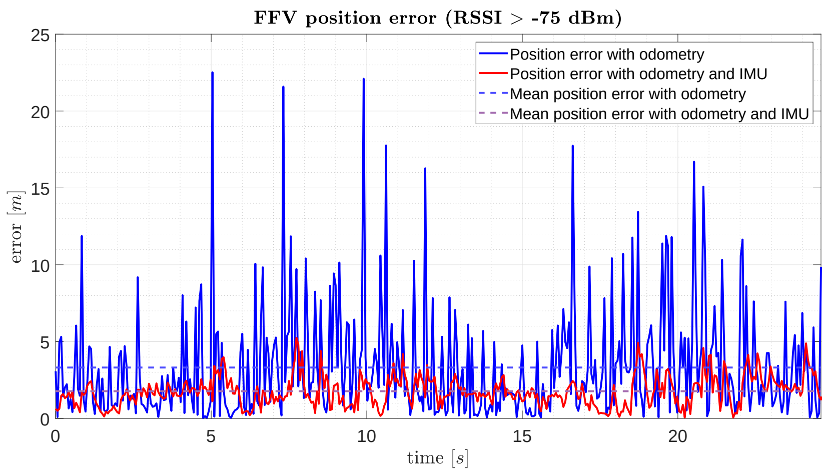
Figure 8.13: Moving FFV outdoor position (RSSI>-80 [dBm])

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	4.7367	15.0375	3.8778
Odometry and IMU	2.1041	1.2827	1.1325

Table 8.9: Mean, variance and std. dev. of the FFV position error (RSSI>-80 [dBm])



(a) Moving FFV outdoor position with odometry and odometry + IMU



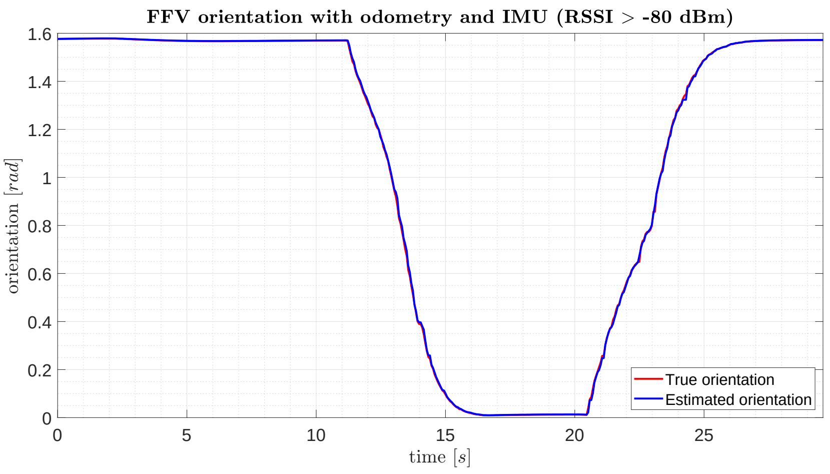
(b) Stationary FFV outdoor position error with odometry and odometry + IMU

Figure 8.14: Moving FFV outdoor position (RSSI>-75 [dBm])

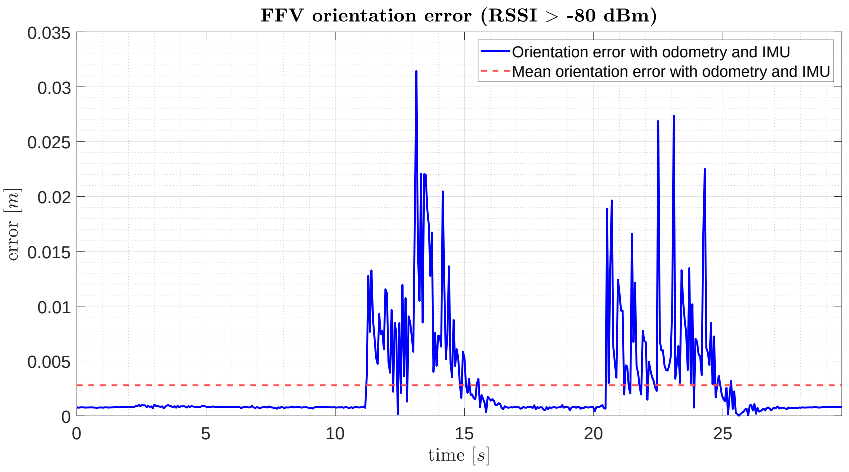
	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry	3.3154	12.2431	3.4990
Odometry and IMU	1.7739	0.9569	0.9782

Table 8.10: Mean, variance and std. dev. of the FFV position error (RSSI>-75 [dBm])

Drawing inspiration from Figure 8.13 and 8.14, the estimated position retrieved using the sensor fusion technique is better than the one obtained using only the odometry information: the green plot is more accurate, stable and smooth than the blue one. Furthermore, as depicted in Table 8.9 and 8.10, the mean error obtained with the "RSSI>-75 [dBm]" simulation is lower than the simulation with the RSSI threshold equal to -80 [dBm]. Hence, signal degradation is an important parameter to take into account and to set properly in order to increase accuracy performance of the Wi-Fi localization. Note that, a strict RSSI threshold can cause localization failure due to the fact that it's impossible to detect at least 4 transmitters.
However, speaking about indoor-outdoor performances comparison, the mean position error obtained in the moving outdoor localization, is very high with respect to the one computed in the indoor moving simulation. This means that the RSSI-based Wi-Fi outdoor localization, using both odometry and sensor fusion technique are not suitable and reliable to retrieve the vehicle pose with good accuracy. Moreover, navigation process, which is based on the localization information, can results unstable and unpredictable, causing security problems both regarding the aluminum smelter plant and the vehicle itself.
As depicted in Figure 8.15 and 8.16, the estimated orientation is similar to the true one. The blue line related to the estimated orientation almost overlap the red one related to the true orientation. However, drawing inspiration from Figure 8.11 and 8.12, the orientation error rise up in the time interval $[9 - 13]$ [s] and $[17 - 22]$ [s] which correspond to period in which the vehicle performs right and left turn, respectively. This increasing error orientation can probably cause by noise attached to the gyroscope of the IMU sensor (remind that the inertial measurement sensor is ideal which means that no accelerometer bias affect the orientation measure). Nevertheless, the maximum value of the orientation error is about 0.032 [rad] ($1.83°$), which can be considered negligible with respect to the one retrieved in the moving indoor localization (Table 8.4).



(a) Moving FFV outdoor orientation with odometry and odometry + IMU

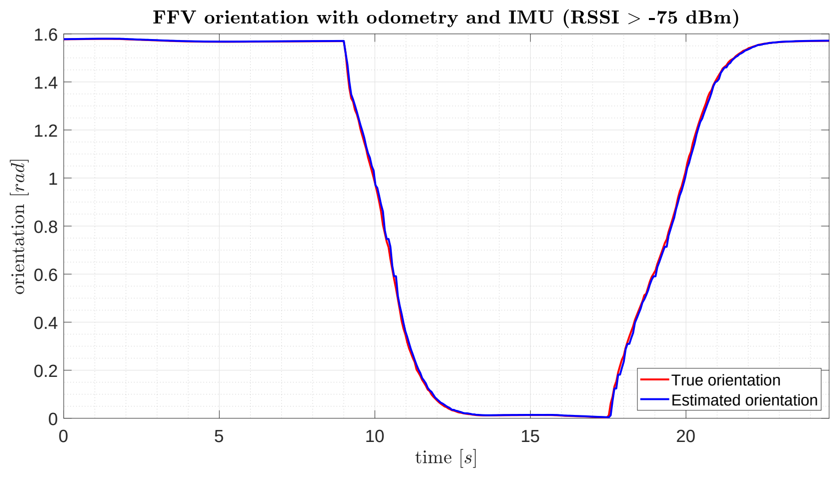


(b) Stationary FFV outdoor orientation error with odometry and odometry + IMU

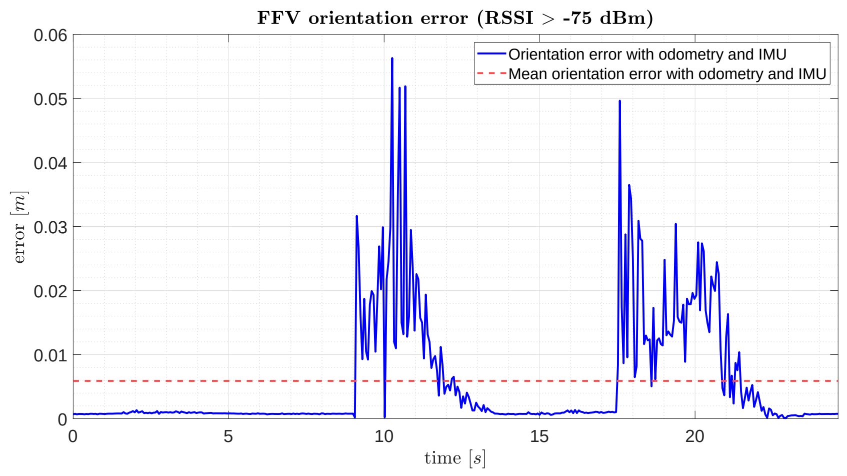
Figure 8.15: Moving FFV outdoor orientation (RSSI>-80 [dBm]): starting point is set to $(x, y) = (-37, -80)[m]$

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry and IMU	0.0028	1.9186×10^{-5}	0.0044

Table 8.11: Mean, variance and std. dev. of the FFV orientation error (RSSI>-80 [dBm])



(a) Moving FFV outdoor orientation with odometry and odometry + IMU



(b) Stationary FFV outdoor orientation error with odometry and odometry + IMU

Figure 8.16: Moving FFV outdoor orientation (RSSI>-75 [dBm]): starting point is set to $(x, y) = (-37, -80)[m]$

	Mean Error [m]	*Variance [m]*	*Std. Dev. [m]*
Odometry and IMU	0.0059	9.2748×10^{-5}	0.0096

Table 8.12: Mean, variance and std. dev. of the FFV orientation error (RSSI>-75 [dBm])

In conclusion, RSSI-based Wi-Fi outdoor localization gives rough estimated FFV position with high mean error, meaning it's not usable for high precision AGV outdoor localization. Moreover with the ideal IMU sensor selected for the thesis simulations, it's impossible to understand how good the estimated orientation value can be and how it affects the estimated pose of the FFV using the sensor fusion technique. To increase the localization performances, an hybrid localization system can be adopted (Section 1.2): laser scanner or visual cameras can be employed to retrieve more information about the surrounding environments like obstacles or street lines. The latter can be used to determine the vehicle position with respect to the street borders and this is useful to maintain the vehicle near to the center of the road.

9 Conclusion and Future Work

9.1 Conclusion

AGVs, in real world, needs to challenge with many crucial navigation aspects such as path planning, obstacle avoidance, and localization.
This thesis proposes two localization techniques for the aluminum smelter fluoride feeder operative vehicle. The first one, the indoor localization, is performed using the ARTag fiducial markers system, while, the second one, the outdoor localization, is implemented with the RSSI-based Wi-Fi approach. Both localization techniques are tested in a simulated Gazebo environment using a simplified fluoride feeder model defined starting from known information about the vehicle. Moreover, for each localization, stationary and moving simulations are performed: in particular, for the moving test, the FFV had to follow a "S" path in order to analyze the localization performances in presence of curves.
Considering the indoor fiducial markers localization, the experimental results, using the moving approach, are promising: position and orientation mean error is, respectively, of 0.3046 $[m]$ and $1.45°$. On the other hand, the outdoor RSSI-based Wi-Fi localization accu-

racy performances obtained in the moving simulation are very poor: position and orientation mean error is of 2.1041 $[m]$ and $0.33°$, respectively. Remind that, the great orientation value obtained in the latter simulation is retrieved using an high accuracy IMU sensor with no accelerometer bias and with small gyroscope measurement error: in real world, IMU data are much more noisy affecting both position and orientation error obtained in the outdoor localization.
In conclusion, only the fiducial markers localization technique can be chosen as a valid localization approach, while the RSSI-based Wi-Fi localization, because of its low accuracy performance, needs to be complemented by other localization support (maps, lasers or visual cameras).

9.2 Future work

Future work can be split in two categories: localization techniques improvements and localization validation in real world scenario.
To improve both localization approaches analyzed in this thesis, laser scanners and additional visual cameras can be taken into account to retrieve information about the surrounding environment (obstacles, building structures position, etc.) and to detect natural feature (road lines, doors, etc.).
To validate the aforementioned localization techniques and in particular to test their robustness, real world simulation needs to be performed: indeed, in virtual simulations, many environmental parameters like weather conditions, magnetic field and high temperature are not considered because not supported by the Gazebo simulator. Moreover, a localization systems comparison considering characteristics such as pose accuracy, operating limits of the localization devices with respect to the environmental conditions and installation and maintenance costs, is required.

References

[1] S. G. M. Hossain, H. Jamil, M. Y. Ali, and M. Zahurul Haq, *Automated guided vehicles for industrial logistics - Development of intelligent prototypes using appropriate technology*, 2010, vol. 5.

[2] D. Ronzoni, R. Olmi, C. Secchi, and C. Fantuzzi, *AGV global localization using indistinguishable artificial landmarks*, 2011.

[3] E. Balomenos, C. Kemper, P. Diamantopoulos, D. Panias, I. Paspaliaris, B. Friedrich, *Novel Technologies For Enhanced Energy And Exergy Efficiencies In Primary Aluminium Production Industry*, 2013.

[4] R. Bemthuis, *Development of a planning and control strategy for AGVs in the primary Aluminium industry*, 2017.

[5] W. E. Haupin, *Electrochemistry of the Hall-Heroult process for aluminum smelting*. American Chemical Society, 1983, vol. 60, no. 4.

[6] A. Gil, *Management of the Salt Cake from Secondary Aluminum Fusion Processes*. American Chemical Society, 2005, vol. 44, no. 23.

[7] Thomas Petry, Peter Schmid, Christian Schlatter, *Airborne exposure to polycyclic aromatic hydrocarbons and urinary exrection of 1-hydroxypyrene of carbon anode plant workers*, 1996, vol. 40, no. 3.

[8] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, 01 2003.

[9] P. Sturm, S. Ramalingam, J.P. Tardif, S. Gasparini, J. Barreto, *Camera models and fundamental concepts used in geometric computer vision*. Now Publishers, 2011, vol. 6, no. 1-2.

[10] B. Chen and B. Pan, *Camera calibration using synthetic random speckle pattern and digital image correlation*, 2020, vol. 126.

[11] B. Benligiray, C. Topal, and C. Akinlar, *STag: A stable fiducial marker system*, 2019, vol. 89.

[12] M. Fiala, *Designing Highly Reliable Fiducial Markers*, 2010, vol. 32, no. 7.

[13] A. Zakiev, K. Shabalina, T. Tsoy, and E. Magid, *Pilot Virtual Experiments on ArUco and ArTag Systems Comparison for Fiducial Marker Rotation Resistance*, 2020.

[14] M. Fiala, *ARTag, a fiducial marker system using digital techniques*, 2005, vol. 2, no. 10.

[15] A. Sagitov, K. Shabalina, R. Lavrenov, and E. Magid, *Comparing fiducial marker systems in the presence of occlusion*, 2017, vol. 2017 International Conference on Mechanical, System and Control Engineering (ICMSC).

[16] S. Siltanen, *Theory and applications of marker based augmented reality*, 01 2012.

[17] P. Corke, *Robotics, Vision and Control. Fundamentalalgorithmsin Matlab*. Springer, 2011.

[18] E. Olson, *AprilTag: A robust and flexible visual fiducial system*, 2011, vol. 2011 IEEE International Conference on Robotics and Automation.

[19] A. Tam, H. Shen, J. Liu, and X. Tang, *Quadrilateral Signboard Detection and Text Extraction.*, 01 2003.

[20] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, *Automatic generation and detection of highly reliable fiducial markers under occlusion*, 06 2014, vol. 47.

[21] Yanbin Zhang and Qi Yuan, *A multiple bits error correction method based on cyclic redundancy check codes*, 2008.

[22] C. Zhao and B. Wang, *A MLE-PSO indoor localization algorithm based on RSSI*, 2017.

[23] O. G. Adewumi, K. Djouani, and A. M. Kurien, “Rssi based indoor and outdoor distance estimation for localization in wsn,” 2013, pp. 1534–1539.

[24] M. N. Rahman, M. T. I. A. T. Hanuranto, and S. T. M. T. R. Mayasari, “Trilateration and iterative multilateration algorithm for localization schemes on wireless sensor network,” pp. 88–92, 2017.

[25] V. Sangale and A. Shendre, *Localization of a Mobile Autonomous Robot Using Extended Kalman Filter*, 2013.

Acknowledgments

I want to thank Prof. Cenedese to give me the opportunity to work on this project, pushing me up and give me fundamental advices during all over these six months.

Techmo Car S.P.A. and, in particular, Eng. Mauro Schiavo for their availability, professionalism and dedication.

Matteo B. and Claudio V. to have been helping each other, especially on the difficult periods faced during our thesis.

Luca B. and Luca T. for all their support during the last two years of the master's degree, especially for the "do it yourself!" and "OK guys!" presentations.

All my friends to share with me all my beautiful life moments and this important goal.

My beautiful family, in particular my mom, my dad and my brother, to support and to believing in me during all these years, pushing me further my limits.

Last but not least, a special thank you goes to Michela. I'm here thanks of you. You are my everything and i love you so much.

A

Markers and Wi-Fi transmitters location

In this chapter, the pose of the fiducial markers and the position of the Wi-Fi transmitters used, respectively, for the indoor and outdoor FFV localization are depicted. Note that, due to the wireless omnidirectional antennas model, only the position of the transmitters are defined. Moreover, for both the localization, different tables are defined for stationary and moving localization simulation setup. The first three tables describe the position of the wireless antennas on the stationary outdoor localization, while the fourth one highlight the transmitters position during the moving test. The final tables depict position and orientation of the fiducial markers for both stationary and moving simulations.

Wi-Fi Transmitters Position			
Transmitters	$\mathbf{T}^{\text{world}}_{\text{transmitter}}$	***Transmitters***	$\mathbf{T}^{\text{world}}_{\text{transmitter}}$
0	$\begin{bmatrix} -45 & -90 & 5 \end{bmatrix}^T$	2	$\begin{bmatrix} -29 & -100 & 5 \end{bmatrix}^T$
1	$\begin{bmatrix} -45 & -50 & 5 \end{bmatrix}^T$	3	$\begin{bmatrix} -19 & -60 & 5 \end{bmatrix}^T$

Table A.1: Wi-Fi transmitters position for stationary FFV simulation with $T_x = 4$

Wi-Fi Transmitters Position			
Transmitters	$\mathbf{T}_{\mathbf{transmitter}}^{\mathbf{world}}$	*Transmitters*	$\mathbf{T}_{\mathbf{transmitter}}^{\mathbf{world}}$
0	$\begin{bmatrix}-45 & -90 & 5\end{bmatrix}^T$	4	$\begin{bmatrix}-65 & -90 & 5\end{bmatrix}^T$
1	$\begin{bmatrix}-45 & -50 & 5\end{bmatrix}^T$	5	$\begin{bmatrix}-55 & -70 & 5\end{bmatrix}^T$
2	$\begin{bmatrix}-29 & -100 & 5\end{bmatrix}^T$	6	$\begin{bmatrix}-37 & -70 & 5\end{bmatrix}^T$
3	$\begin{bmatrix}-19 & -60 & 5\end{bmatrix}^T$	7	$\begin{bmatrix}-19 & -80 & 5\end{bmatrix}^T$

Table A.2: Wi-Fi transmitters position for stationary FFV simulation with $T_x = 8$

Wi-Fi Transmitters Position			
Transmitters	$\mathbf{T}_{\mathbf{transmitter}}^{\mathbf{world}}$	*Transmitters*	$\mathbf{T}_{\mathbf{transmitter}}^{\mathbf{world}}$
0	$\begin{bmatrix}-45 & -90 & 5\end{bmatrix}^T$	7	$\begin{bmatrix}-19 & -80 & 5\end{bmatrix}^T$
1	$\begin{bmatrix}-45 & -50 & 5\end{bmatrix}^T$	8	$\begin{bmatrix}-55 & -110 & 5\end{bmatrix}^T$
2	$\begin{bmatrix}-29 & -100 & 5\end{bmatrix}^T$	9	$\begin{bmatrix}-45 & -100 & 5\end{bmatrix}^T$
3	$\begin{bmatrix}-19 & -60 & 5\end{bmatrix}^T$	10	$\begin{bmatrix}-37 & -110 & 5\end{bmatrix}^T$
4	$\begin{bmatrix}-65 & -90 & 5\end{bmatrix}^T$	11	$\begin{bmatrix}-19 & -90 & 5\end{bmatrix}^T$
5	$\begin{bmatrix}-55 & -70 & 5\end{bmatrix}^T$	12	$\begin{bmatrix}-55 & -60 & 5\end{bmatrix}^T$
6	$\begin{bmatrix}-37 & -70 & 5\end{bmatrix}^T$	13	$\begin{bmatrix}-29 & -50 & 5\end{bmatrix}^T$

Table A.3: Wi-Fi transmitters position for stationary FFV simulation with $T_x = 14$

Wi-Fi Transmitters Position			
Transmitters	$\mathbf{T}^{\text{world}}_{\text{transmitter}}$	*Transmitters*	$\mathbf{T}^{\text{world}}_{\text{transmitter}}$
0	$\begin{bmatrix}-42 & -90 & 5\end{bmatrix}^T$	7	$\begin{bmatrix}-42 & -90 & 5\end{bmatrix}^T$
1	$\begin{bmatrix}-31 & -80 & 5\end{bmatrix}^T$	8	$\begin{bmatrix}0.0 & -50 & 5\end{bmatrix}^T$
2	$\begin{bmatrix}-42 & -70 & 5\end{bmatrix}^T$	9	$\begin{bmatrix}9.0 & -30 & 5\end{bmatrix}^T$
3	$\begin{bmatrix}-31 & -60 & 5\end{bmatrix}^T$	10	$\begin{bmatrix}19 & -50 & 5\end{bmatrix}^T$
4	$\begin{bmatrix}-42 & -50 & 5\end{bmatrix}^T$	11	$\begin{bmatrix}19 & -40 & 5\end{bmatrix}^T$
5	$\begin{bmatrix}-31 & -40 & 5\end{bmatrix}^T$	12	$\begin{bmatrix}19 & -20 & 5\end{bmatrix}^T$
6	$\begin{bmatrix}-20 & -50 & 5\end{bmatrix}^T$	13	$\begin{bmatrix}9.0 & -10 & 5\end{bmatrix}^T$

Table A.4: Wi-Fi transmitters position for moving FFV simulation

ARTag Markers Position and Orientation					
ID	$\mathbf{R}^{\text{world}}_{\text{tag}}$	$\mathbf{T}^{\text{world}}_{\text{tag}}$	*ID*	$\mathbf{R}^{\text{world}}_{\text{tag}}$	$\mathbf{T}^{\text{world}}_{\text{tag}}$
0	$\begin{bmatrix}0 & 0 & -1\\ -1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 25 & 3.5\end{bmatrix}^T$	5	$\begin{bmatrix}0 & -1 & 0\\ 0 & 0 & 1\\ -1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}75 & 0.5 & 3.5\end{bmatrix}^T$
1	$\begin{bmatrix}0 & 0 & -1\\ -1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 50 & 3.5\end{bmatrix}^T$	6	$\begin{bmatrix}0 & -1 & 0\\ 0 & 0 & 1\\ -1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}85 & 0.5 & 3.5\end{bmatrix}^T$
2	$\begin{bmatrix}0 & 0 & -1\\ 0 & -1 & 0\\ -1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 80 & 3.5\end{bmatrix}^T$	7	$\begin{bmatrix}0 & -1 & 0\\ 0 & 0 & 1\\ -1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}95 & 0.5 & 3.5\end{bmatrix}^T$
3	$\begin{bmatrix}0 & 0 & -1\\ -1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 20 & 3.5\end{bmatrix}^T$	8	$\begin{bmatrix}0 & 0 & -1\\ 0 & -1 & 0\\ -1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 60 & 3.5\end{bmatrix}^T$
4	$\begin{bmatrix}0 & 0 & -1\\ -1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 40 & 3.5\end{bmatrix}^T$	9	$\begin{bmatrix}0 & 0 & -1\\ -1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}100.2 & 15 & 3.5\end{bmatrix}^T$

Table A.5: ARTag markers pose from ID 0 to ID 9

ARTag Markers Position and Orientation					
ID	$\mathbf{R}_{\mathbf{tag}}^{\mathbf{world}}$	$\mathbf{T}_{\mathbf{tag}}^{\mathbf{world}}$	*ID*	$\mathbf{R}_{\mathbf{tag}}^{\mathbf{world}}$	$\mathbf{T}_{\mathbf{tag}}^{\mathbf{world}}$
10	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 100.2 & 70 & 3.5 \end{bmatrix}^T$	21	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 65 & 3.5 \end{bmatrix}^T$
11	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 95 & 100 & 3.5 \end{bmatrix}^T$	22	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 75 & 3.5 \end{bmatrix}^T$
12	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 75 & 24 & 3.5 \end{bmatrix}^T$	23	$\begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 100.2 & 90 & 3.5 \end{bmatrix}^T$
13	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 35 & 3.5 \end{bmatrix}^T$	24	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 25 & 3.5 \end{bmatrix}^T$
14	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 85 & 24 & 3.5 \end{bmatrix}^T$	25	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 101 & 24 & 3.5 \end{bmatrix}^T$
15	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 85 & 3.5 \end{bmatrix}^T$	26	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 105 & 109 & 3.5 \end{bmatrix}^T$
16	$\begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 100.2 & 30 & 3.5 \end{bmatrix}^T$	27	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 115 & 109 & 3.5 \end{bmatrix}^T$
17	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 85 & 3.5 \end{bmatrix}^T$	28	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 125 & 109 & 3.5 \end{bmatrix}^T$
18	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 95 & 24 & 4.5 \end{bmatrix}^T$	29	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 105 & 85.2 & 3.5 \end{bmatrix}^T$
20	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 91 & 55 & 3.5 \end{bmatrix}^T$			

Table A.6: ARTag markers pose from ID 10 to ID 29

B
Python scripts

To perform suitable simulations, Gazebo simulator and ROS provide plugins, topics and nodes to obtain data and to better understand the performance of the indoor and outdoor localization tests. However, these information are not sufficient to compute the position and orientation of the fluoride feeder vehicle in the simulation environment. For example, in the indoor localization, the ROS node *ar_track_alvar*, which is used to detect markers (Appendix C), gives us only the 3D distance between tags and camera, while, in the outdoor localization, the *feeder_odometry*, which is used to compute the RSSI value (equation 5.4), gives us only the odometry information about the position of the fluoride feeder vehicle with respect to the 3D world reference frame. Thus, extrapolated data needs to be elaborated using suitable and efficient algorithms (scripts) in order to obtain information about position and orientation of the vehicle with respect to the 3D world coordinates. These scripts are created using the *Python* high-level programming language, which is, nowadays, one of the most used programming language in the world.

For this project, two main scripts have been implemented:

1. *marker_localization*, which is the algorithm used for retrieve the indoor FFV position and orientation with respect to the 3D world reference frame

2. *wifi_localization*, which is the script used for compute the outdoor FFV position with respect to the 3D world coordinates

B.1 Marker_localization script

```
#!/usr/bin/env python

###################################################################
######## LOAD PYTHON PACKAGES AND DEFINE GLOBAL VARIABLES #########
###################################################################

def marker_detection(msg):
  if len(msg.markers) > 0:
    detection_pose = msg.markers[0].pose
    i = msg.markers[0].i
    name = msg.markers[0].header.frame_id

    if i != 255 and i < numTags:
      if name=="/camera_right_link":
        translation_camera = trans_cam_right
        R_camera = R_cr
      elif name=="/camera_left_link":
        translation_camera = trans_cam_left
        R_camera = R_cl
      elif name=="/camera_front_link":
        translation_camera = trans_cam_front
        R_camera = R_cf

        if i==0 or i==1 or i==4 or i==9 or i==3 or i==16 or i==23:
          R = R_0
        if i==2 or i==8 or i==10:
          R = R_1
        if i==29 or i==30:
          R = R_2
        if i==5 or i==6 or i==7:
          R =  R_3
        if i==12:
          R = R_4
        if i==11:
          R = R_5
        if i==13 or i==15 or i==17 or i==20 or i==21 or i==22 or i==24:
          R = R_6
        if i==14 or i==18 or i==26 or i==27 or i==28 or i==25:
          R = R_7

        # from camera distance on vehicle reference frame to the camera one
        trans_camera = np.transpose(R_camera).dot(translation_camera)
        # from roto translation on the camera frame to roto translation on tag frame
        trans = [detection_pose.pose.position.x, detection_pose.pose.position.y,
                detection_pose.pose.position.z]
        trans_2 = trans + trans_camera
        rot = [detection_pose.pose.orientation.x, detection_pose.pose.orientation.y,
               detection_pose.pose.orientation.z, detection_pose.pose.orientation.w]
        transform = tf.transformations.concatenate_matrices(
                    tf.transformations.translation_matrix(trans_2),
                    tf.transformations.quaternion_matrix(rot))
        inversed_transform = tf.transformations.inverse_matrix(transform)
        translation = tf.transformations.translation_from_matrix(inversed_transform)
        #camera rotation with respect to the tag reference frame
        quaternion = tf.transformations.quaternion_from_matrix(inversed_transform)
        (r_1,p_1,y_1) = tf.transformations.euler_from_quaternion(quaternion)
        quaternion_matrix = quaternion_matrix_function(quaternion)
        Rot_vehicle_world = R.dot(quaternion_matrix.dot(np.transpose(R_camera)))
```

```
      q = quaternion_from_matrix(Rot_vehicle_world)

      if abs(translation[0]) < limit_dist and abs(translation[1]) < limit_dist and
         abs(translation[2]) < limit_dist and abs(p_1) < limit_angle:
        pose_vehicle = p_tags[:,i] + R.dot(translation)
        odom.pose.pose.position.x = pose_vehicle[0]
        odom.pose.pose.position.y = pose_vehicle[1]
        odom.pose.pose.position.z = pose_vehicle[2]
        odom.pose.pose.orientation.x = q[0]
        odom.pose.pose.orientation.y = q[1]
        odom.pose.pose.orientation.z = q[2]
        odom.pose.pose.orientation.w = q[3]
        odom.pose.covariance = odom_covariance
        odom.twist.covariance = odom_covariance

def quaternion_from_matrix(q):
  w = math.sqrt(1 + q[0,0] + q[1,1] + q[2,2]) * 0.5
  x = (q[2,1] - q[1,2]) / (4*w)
  y = (q[0,2] - q[2,0]) / (4*w)
  z = (q[1,0] - q[0,1]) / (4*w)
  quat = np.array([x,y,z,w])
  return quat

def quaternion_matrix_function(q):
  q_matr = np.zeros((3,3))
  q_matr[0,0] = 1 - 2*math.pow(q[1],2) - 2*math.pow(q[2],2)
  q_matr[0,1] = 2*q[0]*q[1] - 2*q[2]*q[3]
  q_matr[0,2] = 2*q[0]*q[2] + 2*q[1]*q[3]
  q_matr[1,0] = 2*q[0]*q[1] + 2*q[2]*q[3]
  q_matr[1,1] = 1 - 2*math.pow(q[0],2) - 2*math.pow(q[2],2)
  q_matr[1,2] = 2*q[1]*q[2] - 2*q[0]*q[3]
  q_matr[2,0] = 2*q[0]*q[2] - 2*q[1]*q[3]
  q_matr[2,1] = 2*q[1]*q[2] - 2*q[0]*q[3]
  q_matr[2,2] = 1 - 2*math.pow(q[0],2) - 2*math.pow(q[1],2)
  return q_matr
```

```
r = rospy.Rate(15)
marker_sub = rospy.Subscriber("/ar_pose_marker",AlvarMarkers,marker_detection)
marker_pub = rospy.Publisher("/vehicle_localization", Odometry, queue_size = 10)

while not rospy.is_shutdown():
  odom.header.stamp = rospy.Time.now()
  odom.header.seq = seq
  seq = seq+1

  if odom.pose.pose.orientation.x!=0 and odom.pose.pose.orientation.y!=0
     and odom.pose.pose.orientation.z!=0 and odom.pose.pose.orientation.w!=0:
    odom_broadcaster.sendTransform((odom.pose.pose.position.x,
            odom.pose.pose.position.y,0.35),(odom.pose.pose.orientation.x,
            odom.pose.pose.orientation.y,odom.pose.pose.orientation.z,
            odom.pose.pose.orientation.w), rospy.Time.now(),"base_link","odom")

  marker_pub.publish(odom)
  r.sleep()
```

Listing B.1: Marker localization algorithm

B.2 Wifi_localization script

```
#!/usr/bin/env python

###################################################################
######## LOAD PYTHON PACKAGES AND DEFINE GLOBAL VARIABLES #########
###################################################################

def callback(msg):

  rssi = np.zeros((num_trans))
  (rssi, index) = findRssi(msg.position)
  rssi = np.asarray(rssi)
  l = len(index)
  index2 = np.asarray(index)
  rospy.logwarn(str(len(index2)))
  dist = np.zeros((1,l))
  A = np.zeros((l-1,3))
  b = np.zeros((l-1,1))
  j = 0

  if len(index2)>=4:
    for i in range(l):
      dist[0,i] = math.pow(10,(rssi[index[i]] - tx_pow - tx_gain - rx_gain -
                  20*math.log10(wave)+20*math.log10(4*math.pi))/(-10*attenuation))

    for i in range(l-1):
      A[i,0] = position_trans[index[i+1],0] - position_trans[index[0],0]
      A[i,1] = position_trans[index[i+1],1] - position_trans[index[0],1]
      A[i,2] = position_trans[index[i+1],2] - position_trans[index[0],2]
      b[i,0] = 0.5*(math.pow(dist[0,0],2) - math.pow(dist[0,i+1],2) +
               math.pow(position_trans[index[0],0] -
               position_trans[index[i+1],0],2) +
               math.pow(position_trans[index[0],1] -
               position_trans[index[i+1],1],2) +
              math.pow(position_trans[index[0],2] -
              position_trans[index[i+1],2],2))

    x_1 = np.linalg.pinv(np.transpose(A).dot(A))
    x_2 = x_1.dot(np.transpose(A))
    x_3 = x_2.dot(b[:,0])
    x_ls = x_3 + np.transpose(position_trans[index[0],:])
    odom_pub.pose.pose.position.x = x_ls[0]
    odom_pub.pose.pose.position.y = x_ls[1]
    odom_pub.pose.pose.position.z = x_ls[2]
    odom_pub.pose.pose.orientation.x = 0
    odom_pub.pose.pose.orientation.y = 0
    odom_pub.pose.pose.orientation.z = 0
    odom_pub.pose.pose.orientation.w = 1
    odom_pub.pose.covariance = odom_covariance
    odom_pub.twist.covariance = odom_covariance

def findRssi(x):
  index = []
  a = []
  noise = np.random.normal(0,6,1)
  j = 0
  for i in range(num_trans):
    d = distance(x,i)
    r = tx_pow + tx_gain + rx_gain - noise[0] + 20*math.log10(wave) -
        20*math.log10(4*math.pi) - 10*attenuation * math.log10(d)
    a.append(r)
    if r > -75:
      index.append(i)
  return a,index

def distance(y,index):
    return math.sqrt(math.pow((position_trans[index,0] - y.x),2) +
                     math.pow((position_trans[index,1] - y.y),2) +
                     math.pow((position_trans[index,2] - y.z),2))
```

```

rospy.init_node('wifi_localization')
feeder_sub = rospy.Subscriber('/feeder_odometry',Pose, callback)
wifi_odom_pub = rospy.Publisher('/wifi_odom',Odometry,queue_size = 10)

r = rospy.Rate(15)

while not rospy.is_shutdown():
    odom_pub.header.stamp = rospy.Time.now()
    odom_pub.header.seq = seq
    seq = seq+1
    if odom.pose.pose.orientation.x!=0 and odom.pose.pose.orientation.y!=0
     and odom.pose.pose.orientation.z!=0 and odom.pose.pose.orientation.w!=0:
    odom_broadcaster.sendTransform((odom.pose.pose.position.x,
            odom.pose.pose.position.y,0.35),(odom.pose.pose.orientation.x,
            odom.pose.pose.orientation.y,odom.pose.pose.orientation.z,
            odom.pose.pose.orientation.w), rospy.Time.now(),"base_link","odom")

    wifi_odom_pub.publish(odom_pub)
    r.sleep()
```

Listing B.2: Wifi localization algorithm

C
ROS nodes and topics

In this section, a briefly description of the relevant ROS nodes and topics used for the simulations is given.

Indoor localization

- *individualMarkersNoKinect*: it is used to perform marker detection. It requires as input parameters the marker size, the name of the visual camera and its calibration parameters, the max_new_marker_error which is a threshold determining when new markers can be detected under uncertainty and the max_track_error which is a threshold determining how much tracking error can be observed before an tag is considered to have disappeared. As output, it provides the ID of the detected tag and its translation vector with respect to the camera coordinate frame. It publishes the output data (AlvarMarker custom data type) over the */ar_pose_marker* topic.
- *markerLocalization*: it is used to compute the FFV pose in the world coordinate frame. It takes, as input, the information from the /ar_pose_marker topic and publish the fluoride feeder vehicle pose (odometry data type) over the */vehicle_localization* topic.

Outdoor localization

- *wifiLocalization*: it is used to retrieve the RSSI value starting from the true position of the FFV, obtained subscribing the */feeder_odometry* topic, and the transmitters. Then, using the trilateration algorithm, the position of the vehicle is computed and published over the odometry data type topic */wifi_odom*.

Indoor and outdoor localization

- *feederOdometry*: it is used to retrieve the true position of the fluoride feeder vehicle and publish it over the /feeder_odometry topic.
- *imuCovariance*: it is used to adding the covariance matrix in the */imu* topic and publish the complete IMU data type message over the */imu_cov* topic.
- *ekfLocalization*: it is used to fuse the odometry information acquired from the /vehicle_localization or the /wifi_odom topics with the inertial motion unit data obtained from the /imu_cov topic. The ekfLocalization output is an odometry data type message published over the */odometry_filtered topic*.
- *simpleMove*: it is used to move the FFV on the Gazebo environment. It requires, as input parameter, the initial position of the vehicle, the velocity in [*km/h*] to achieve and, if the trajectory is not linear, the turn angle and its direction. The output is published over the */ackermann_cmd* topic.

ROS computational graph for indoor and outdoor localization are depicted in Figure C.1 and Figure C.2. Note that, all the nodes and topics not described above but present in the ROS graphs are internal to the ROS framework and useful for the simulations. Their functionality and nature description are not required for the thesis purposes.

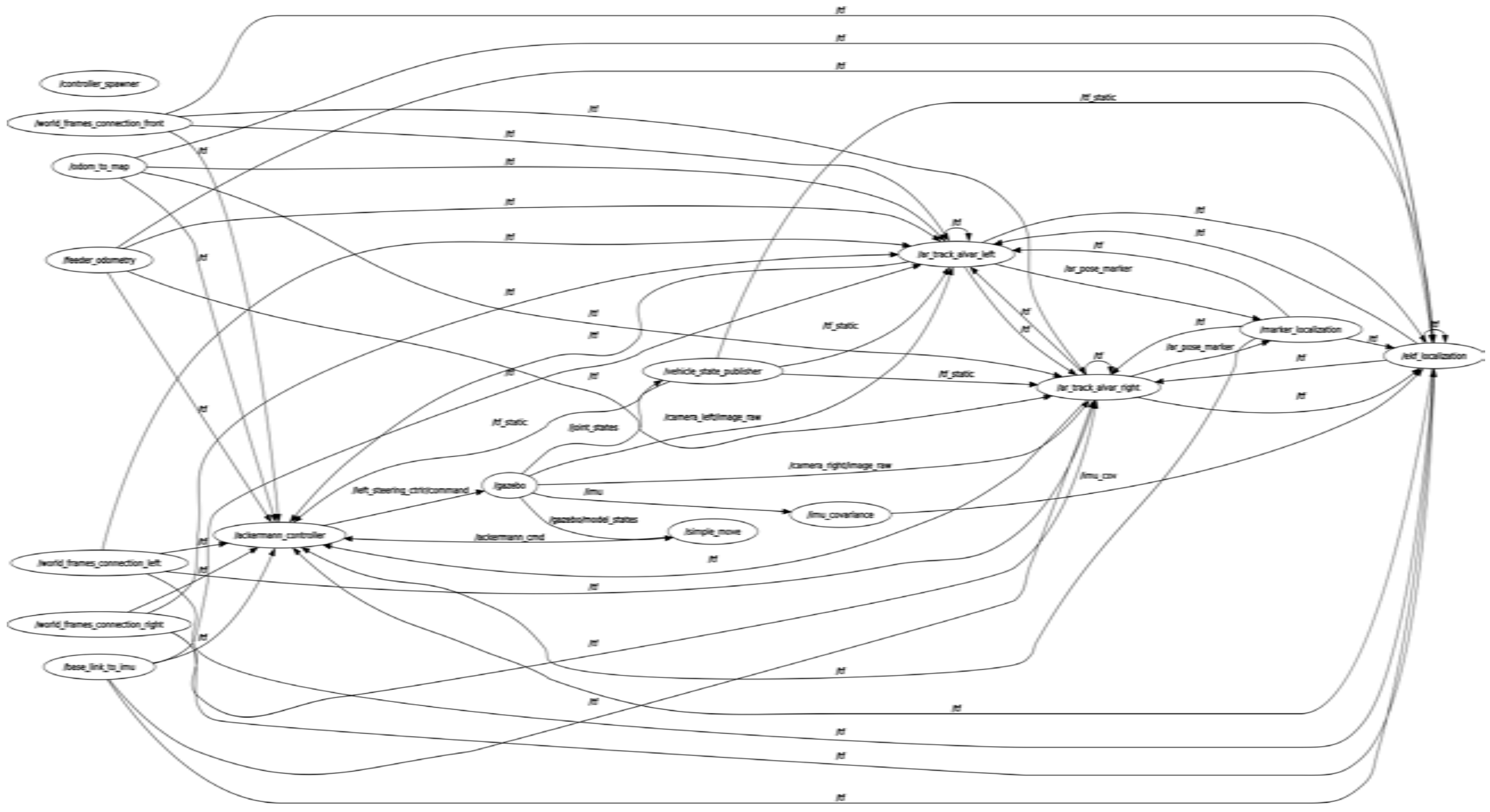


Figure C.1: ROS computational graph for the indoor localization

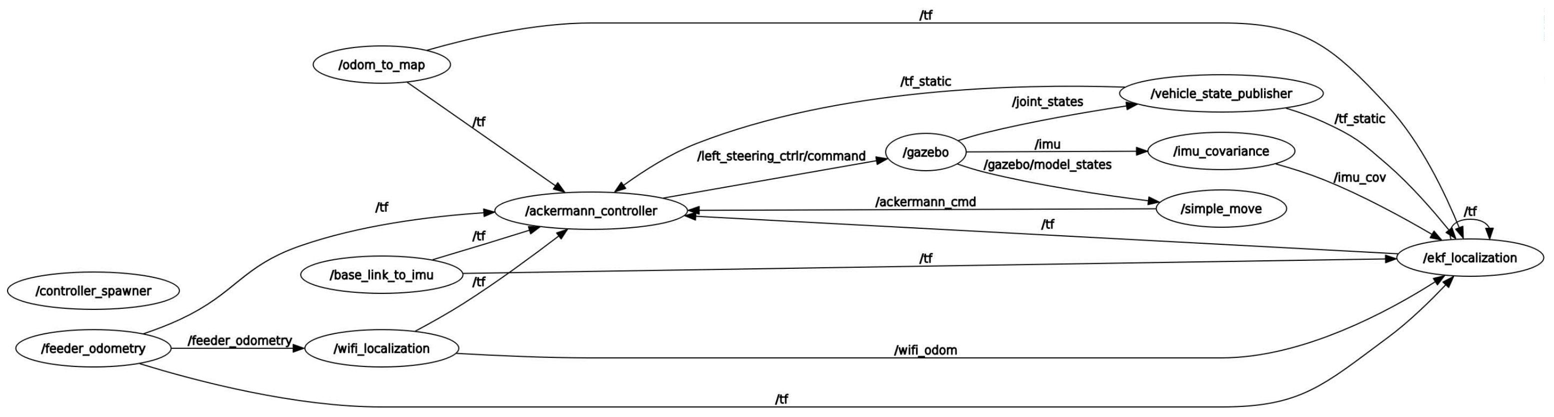


Figure C.2: ROS computational graph for the outdoor localization