POLITECNICO DI TORINO

Master of Science in Mechatronic Engineering

Environment perception and road boundary detection from LiDAR data

Supervisor: prof. Andrea TONOLI

Co-supervisor: prof. Nicola AMATI

prof. Angelo BONFITTO

Candidate:

LORENZO FULLONE

Identification number: 246862

Academic Year 2018-2019

I want to thanks my family that has supported me for all these years, without them I could never reach this fantastic goal.
To myself and all the sacrifice I've done in these years to finally achieve this important result.
To all my friends who have remained at my side, both in bad times and in the good ones.
To the supervisor, the co-supervisors, Stefano and all the guys I worked with during the thesis period that helped me get to the end.

Abstract

Self-driving cars are in continuous development and they will give us a safer and most powerful way of moving. The research, in the autonomous vehicles field, is working all the time in order to meet all the requirements in terms of real-time, safety and traffic congestion optimization. To do these, three different development areas are the most crucial: environment perception for mapping and localization, planning and decision-making for the behavioral part and vehicle control in order to give the vehicle the proper input.

The environment perception is one of the critical parts because the car has to know what there is in its surroundings: if there something moving like other cars, bikes or people and if there are some fixed obstacles to avoid the collision with them. This is crucial also because the localization of a vehicle cannot be done using only the Global Positioning System or the Inertia Measurements, thus it becomes more and more challenging to have a really good estimate of the mapping environment with high precision. In this thesis work, I proposed a SLAM approach using data coming from the LiDAR sensor. The main strategy is to use a scan-matching algorithm in order to give to the tracker the transformation between consecutive scans, that computes the relative position and then map the entire car surrounding into a single map with the track of the vehicle trajectory. This is done thanks to the Iterative Closest Point and the Normal Distribution Transform algorithms that are powerful point-set registration processes that gives excellent results in terms of accuracy, compared to other methods, for the estimate of the transformation from a moving scan to a reference one.

The overall SLAM method has been carried out taking a cue from some other techniques and testing time to time different parameters setups: the car can be larger or smaller, the perceived environment can change based on the road we are on and sometimes we need improved real-time response.

The boundary construction problem is strongly related to the environment perception because the car has to know also where it can go without making any damage to things or people, and what the behavioral planner has to command to the controller in order

to make a turn or avoid an obstacle. With my work for this part, the vehicle has a free area in which it can move defined, in fact, through the lines of the boundaries obtained thanks to an interpolation method. The idea is to discretize the processed scan into shapes and then obtain the front free space of the vehicle. This boundary construction process is entirely made through point cloud processing and shapes analyzing.
The overall coding and algorithms have been developed using MATLAB and related toolbox and they have been tested with some experimental data took with a Velodyne VLP-16 LiDAR.

Contents

Chapter 1

Introduction

Self-driving cars are expected to make big changes in our transportation system because they have an important impact on road safety, traffic jam and way of moving. The biggest companies in automotive industry like Tesla, General Motor, Ford and educational laboratories are all working on the research and development of autonomous vehicles. The vehicles have to be controlled using data coming from different sensors and this is one of the most challenging work in the self-driving cars field. Typical sensors used on self-driving cars are lidar, radar, GPS, inertial measurement unit (IMU) and stereo vision camera. Generally, modern self-driving cars use simultaneous localization and mapping (SLAM) algorithms [1] to detect the surrounding environment and other moving objects like vehicles, motorbike, bike and pedestrians. In the R&D process one of the most important phase is to have the more reliable perception of the environment through the sensors measurement, so Robot Operating System (ROS) and MATLAB-Simulink are highly used since engineers and researchers can test their algorithm in simulation and directly on the car.

1.1 Motivation

The main idea coming from the SLAM algorithm is that one can start to navigate and to drive in the proper way only knowing precisely where to go, where it is and how to get to the destination. So, a self-driving car must be able to calculate its own position and orientation with respect to the surrounding environment: this problem is known as ego pose estimation. The necessity of a map is crucial for the environment perception in autonomous driving and that's the reason why the self-driving cars have to be equipped with a lot of powerful sensors.

Cameras are passive sensors that collect light reflected from the environment, but their data are conditioned by the huge world variations (darkness, rain, fog, low sun, snow, ...). Camera data do not provide any range information: if we want it, we need to use multiple cameras (stereo vision) but it's more complex and less reliable as the range increase.

LiDARs (Light Detection And Ranging) [2] are sensors that collect the Euclidean distances, also called ranges, between the sensor location and the points in the surrounding. The LiDAR instrument fires rapid pulses of laser light at a surface and it measures the amount of time each pulse takes to bounce back. Thanks to laser rotation, a 3D point cloud of the environment is built with range high accuracy information up to 150 meter and often with a 360° field of view. The weakness of LiDAR lies in weather condition changes and in high sun angles and reflections.

From the pros and cons of cameras and LiDARs we can see that the strengths of each sensor may be used to compensate for the weaknesses of the other when the two are combined. So, technically we would like the camera's dense angular resolution to compensate for the sparsity of the LiDAR, and the accurate range information of the LiDAR to resolve the ambiguity in camera depth perception.

Instead of LiDARs one can use the radars that are cheaper, but the accuracy of them in both range and angular direction is particularly poor compared to LiDAR. Ergo, radar is widely used in obstacle detection rather than pose estimation.

Others sensors may be used for localization. The Global Positioning System (GPS) [3] provides users with positioning, navigation and timing (PNT) and enables automatic

vehicle location and in-vehicle navigation systems that are widely used throughout the world today. However, the GPS signal is not always reliable and precise, especially in the civilian application where the accuracy is poor and the signal is degraded in some situations. So, an accurate orientation cannot be determined using GPS only.
The Inertial Measurement Unit (IMU) [4] is another sensor widely used for localization: it's based on multi-axis combinations of precision gyroscopes, accelerometers and magnetometers. This unit works by detecting linear acceleration using one or more accelerometers and rotational rate using one or more gyroscopes. Some also include a magnetometer which is commonly used as a heading reference. An IMU can be integrated into GPS based automotive navigation systems, giving the system a dead reckoning capability and the ability to gather as much accurate data as possible about the vehicle's current speed, turn rate, heading, inclination and acceleration.

1.2 SAE standard for ADAS

The Society of Automotive Engineers (SAE) is a globally active professional association and standards developing organization for engineering professionals in various industries. The SAE International, as described in Fig.1.1, released a document [5] that provides a taxonomy with detailed definitions for six levels of driving automation, ranging from no driving automation (level 0) to full driving automation (level 5), in the context of motor vehicles and their operation on roadways. These level definitions, along with additional supporting terms and definitions, can be used to describe the full range of driving automation features equipped on motor vehicles in a functionally consistent and coherent manner.
ADAS stands for Advanced Driver-Assistance Systems and represents the electronic systems that help the vehicle while driving or during parking. They are designed in order to reduce road fatalities by alerting the driver of potential problems or by avoiding collisions with other road entities.

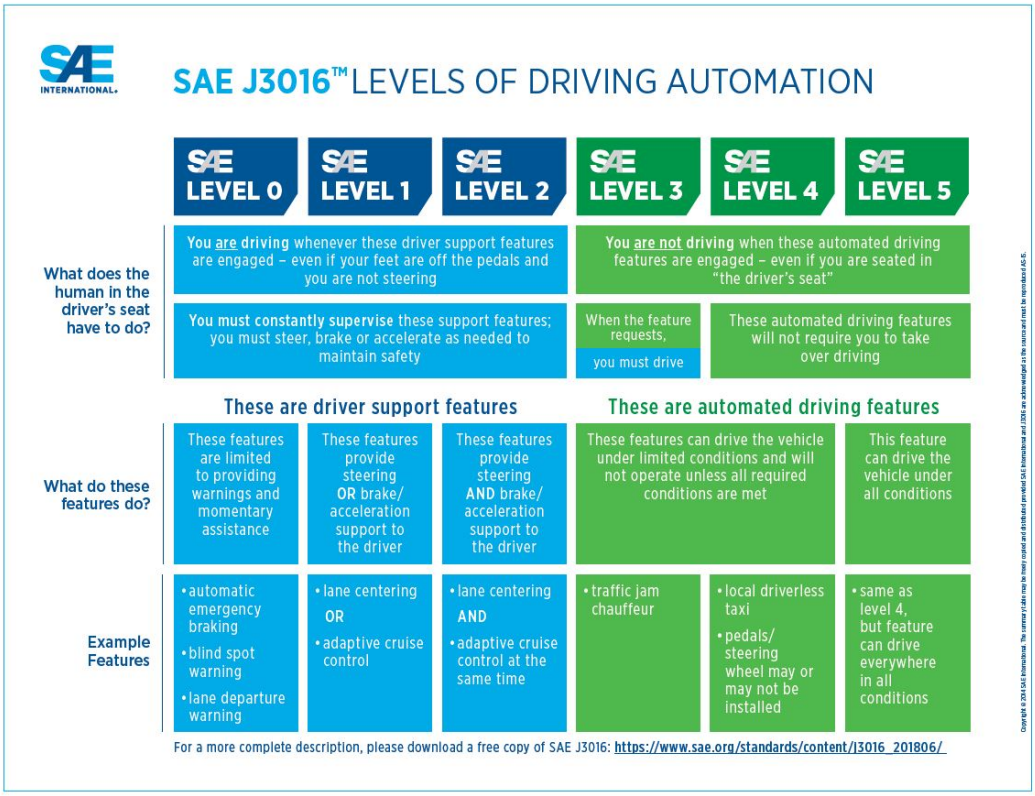


Figure 1.1: *J3016 levels of driving automation.*

So, the six levels can be defined as:

- *Level 0 - No automation :* steering or speed control may be momentarily assisted by the vehicle, but the human driver is in charge of all the aspects of driving;
- *Level 1 - Driver assistance :* longitudinal or lateral support under well-defined driving scenarios(e.g. highway) are guaranteed, because the vehicle takes over either the speed of the steering control on a sustained basis;
- *Level 2 - Partial automation :* both speed and steering control are taken over by the vehicle, therefore continuous longitudinal and lateral support under well-defined driving scenarios are guaranteed. A Level 2 vehicle is equipped with a wider set of ADAS;
- *Level 3 - Conditional automation :* the vehicle becomes capable of taking full control under well-defined driving scenarios, but the driver must be always in the condition of suddenly taking back control when required by the system;

- *Level 4 - High automation :* human interaction is not needed anymore, the vehicle takes full control and complete a journey in full autonomy under limited driving scenarios. Pedals and steering wheel are likely to be still present to guarantee the possibility to drive in scenarios that go beyond the defined uses cases(e.g. off-road);

- *Level 5 - Full automation :* the vehicle takes full control under all driving scenarios, no more provisions for human control are present. The concept of journey will be disruptively innovated, the entire vehicle design revolutionized.

1.3 State of the art

Pose estimation is one of the key technologies for autonomous driving in the urban environment, considering the fact that the GPS is unable to keep high accuracy in urban environments, a very common solution to this problem has emerged in recent years. In an environment that does not have a map, pose transform can be estimated by registration between the front and back frames in which the vehicle's global pose is calculated by the accumulation of the previous transforms, that is odometry or the front-end of SLAM.
A general description of the interacted systems inside a self-driving car can be seen in Fig.1.2. As described in [6] perception refers to the ability of an autonomous system to collect information and extract relevant knowledge from the environment so, "*Environmental perception*" refers to developing a contextual understanding of environment, such as where obstacles are located, detection of road signs/marking, and categorizing data by their semantic meaning. "*Localization*" refers to the ability of the robot to determine its position with respect to the environment.

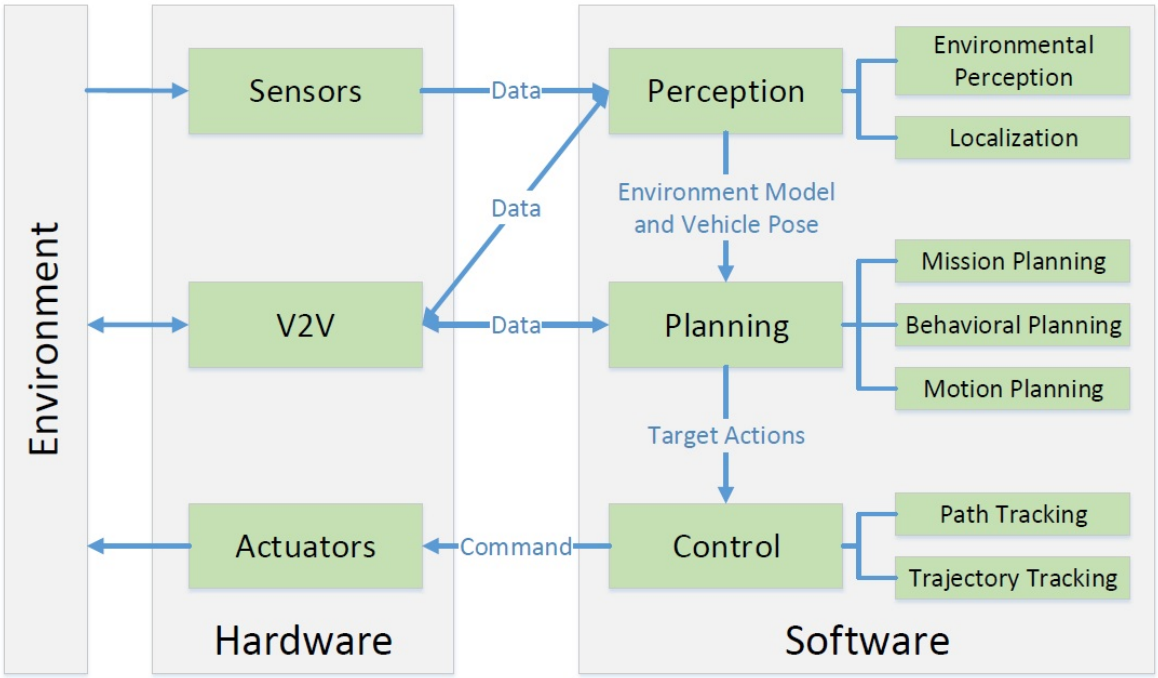


Figure 1.2: *General overview of the Autonomous Vehicle's systems*

1.3.1 Pose Estimation

The SLAM process, as mentioned in [1], where both the trajectory of the platform and the location of the landmarks are estimated on-line without the need for any a priori information of location. A solution to the SLAM problem has been seen as a really important result for the mobile robotics field as it would provide the means to make a robot truly autonomous. SLAM has been formulated and solved as a theoretical problem in several ways and It has been implemented for indoor and outdoor robots, underwater and airborne systems and also in the autonomous driving field.

Over the years a lot of research and work has been done in the map and localization field starting from a probabilistic mapping approach, so the major aspect of this work was to show that there must be a high correlation between estimates of the location of different landmarks in a map and that indeed these correlations would grow with successive observations.

The main architecture of the SLAM, the results and the origin of the name 'SLAM' were first presented in a mobile robotics survey paper presented at the 1995 International Symposium on Robotics Research [7].

In Fig.1.3 it is shown a general description of the SLAM algorithm where there are the

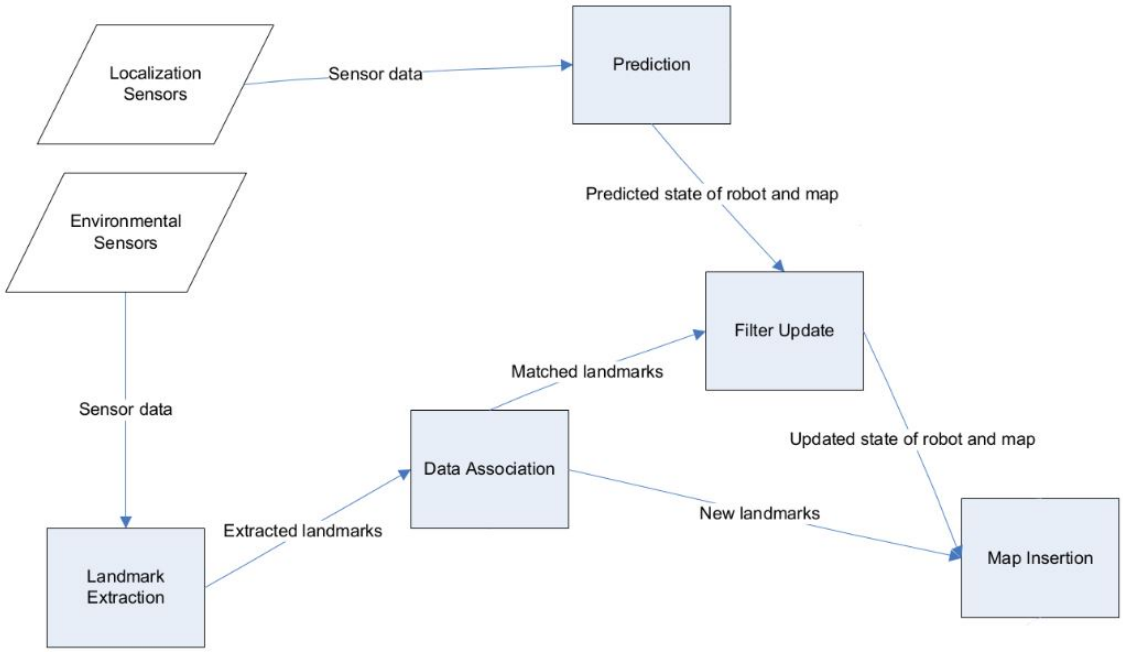


Figure 1.3: *General description of SLAM flowchart*

perception part related to landmarks and the robot state and map estimate; they both contribute to the update of the global map and the robot state.

Another possible approach for the pose estimation is to use the Lidar Odometry And Mapping(LOAM) [8] algorithm, that is similar to the SLAM algorithm, and consists in two different algorithms: the Lidar Odometry algorithm and the Lidar Mapping algorithm. In the Lidar Odometry phase a feature extraction method is implemented to determine edge points and planar points that are used later, in the finding correspondence method, to compute the distances from a feature point to its found correspondence. Then from distances and time stamps the motion is estimated. In the Lidar Mapping phase some data coming from the Lidar Odometry are used to match and register the point-clouds in the world coordinate frame.

One last possible approach to the SLAM problem is described in [9] as Velodyne SLAM that uses only acquired frames to build-up a detailed 3D map i.e. a point cloud of the environment along with the vehicle trajectory. In Fig.1.4 there is a general overview of the method implemented in [9]: first the point cloud of the surrounding is acquired, then is processed in order to give to the localization algorithm proper and valid data; in the localization phase, also called LiDAR motion estimation, is obtained the estimate of the relative pose with respect to the global coordinate frame that is used for the

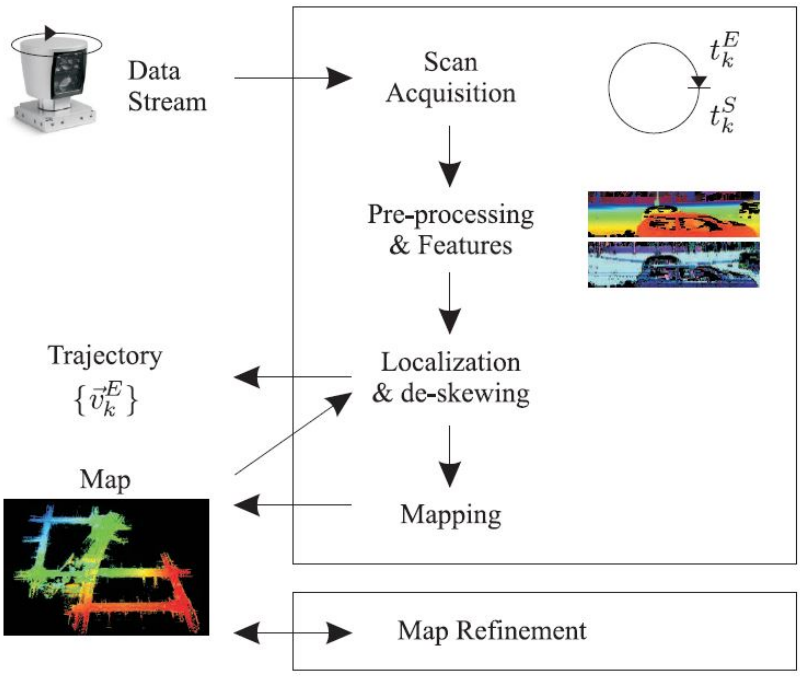


Figure 1.4: *General flowchart of Velodyne SLAM*

mapping phase; the mapping phase consists in a simple map update from the trajectory and the scan points.

LiDAR motion estimation is intended for finding the relative spatial transformation between two frames, based on LiDAR data collected at those frames. Such techniques typically treat the data as sets of 3D points, also known as point clouds. The transformation between two frames is found by aligning the two point clouds using point set registration [10], also known in some cases as scan matching or point matching.

In general, this type of problem is called the point set registration problem, which is one of the most important problems in computer vision and pattern recognition.

Now, the point set registration problems says, given 2 point clouds in two different coordinate frames, and with the knowledge that they correspond to or contain the same object in the world, how shall we align them to determine how the sensor must have moved between two scans? More specifically, we want to figure out the optimal translation and the optimal rotation between the two sensor reference frames that minimize the distance between the two point clouds. The problem is that, in general we don't know which points correspond to each other.

The most popular algorithm for solving this kind of problem is called the Iterative

Closest Point algorithm[10], or ICP for short, and basically this is what it does: given two point set F (fixed) and M (moving) and an initial guess for the transformation it finds iteratively the optimal transformation that aligns the two point clouds by minimizing the square errors between the corresponding entities.
Anyway in the second chapter the main working principle of both ICP and NDT algorithms are presented.

1.3.2 Road boundaries extraction

The road boundary detection and construction problem represents another crucial point in a self-driving car because the car needs to know how the surrounding area is populated in term of vehicles, people, bicycles, motorbikes, trees and all the things that a car can meet in the road. Recently 3D LiDAR has been applied to 3D environment modeling of autonomous vehicles. The 3D LiDARs can provide a large amount of point cloud data with a coverage area of 360°. Therefore it has been widely deployed in autonomous vehicles to participate in the DARPA Grand Challenge[11, 12, 13] and the China Smart City Future Urban Challenge [14, 15, 16] for environment perception. Moreover, a lot of companies have also chosen the 3D LiDAR as an essential sensor to understand the environment surrounding the autonomous vehicle. One of the most widely used methods for detecting road boundaries is to transform the point cloud captured from 3D LiDAR into a grid map, and detect road boundaries based on local geometric features of the grid. Some of the methods already implemented have a strong real-time performance and they are easily blend multiple sensor detection results; however, the grid map resolution is low so the accuracy of the road boundary detection is bad.
Another major road boundary detection method is to detect boundary points based on the LiDAR scanning principle so, in some cases boundary points are detected using a moving window that goes through consecutive scan lines, in other cases the curb-like obstacles are detected by analyzing the ring compression of 3D LiDAR. In these cases, we have more detection accuracy but the real-time performance requirements may not be met because of the high "weight" of the LiDAR data that have to be processed.
In this thesis work, we use 3D LiDAR to detect the environment surrounding the ve-

hicle through the building of a point-map that contain the safe space in which the car can move and the not-free space that can't be reached. So, starting from a dense unorganized set of data points there is a first processing phase where the ground point are eliminated, then a 2D "alpha-shape" extraction phase and then a "rectangular-shape" transformation phase. Really few tests have been done for this part and all with bad preliminary results.

1.4 Thesis organization

The thesis is organized as follows:

- *Chapter 2:* in this chapter the scan-matching algorithm and the approach used for the localization and mapping are described, in particular how ICP algorithm works and what change from this to the other algorithm(NDT). Moreover it's presented the customized boundary construction problem, how it works, how it can be improved and the possibility to integrate it with the localization problem.
- *Chapter 3:* this chapter contain a brief description of the hardware layout of the data acquiring system and a detailed description of the Velodyne LiDAR used for the experimental acquisition. Then the presented methods are tested with different data-sets.
- *Chapter 4:* all the results coming from the tests are commented here.
- *Chapter 5:* in this final chapter, there are conclusions related to the obtained results and some future works to be done are discussed.

Chapter 2

Methods

This chapter contains the detailed description of the localization and mapping algorithm method used, so the main approach description. In general, for real-time applications of algorithms the data load to be processed has to be not so heavy in order to have a low computational cost that is capable to maintain the real-time requirements. On the other hand, data have to be also valid otherwise the functionality of the algorithm is compromised. In the method introduced here, there is a first down-sampling phase that guarantee a good trade-off between the real-time execution time and the data information loss. The purpose of this method is to obtain a map and a pose which represents the real surrounding environment and the real vehicle pose with the maximum possible accuracy.

In the second section there is the description of the boundary construction algorithm implementation that has a similar shape of the one used in the mapping and localization method. For this method the real-time part is not considered crucial and the results coming from that have to be taken as a first approximation.

2.1 Localization and mapping algorithm

In this section the entire process regarding the localization and mapping algorithm is described in detail.

The first presentation of the algorithm is used for processing a file containing all the registered frame, as we can see in Fig.2.1, so it's not the flow used in the real-time application. In real-time there's not an "END?" condition, instead there is an endless loop that continuously process the data coming from the LiDAR sensor.

As seem from the Fig.2.1 there is a pre-processing phase that filters the input frame to gain in computational payload.

Fig.2.1 shows the flowchart of the algorithm:

- *Transformation Initialization*: in this block a first guess for the initial transformation between the first and the next frame is assigned. Usually starting from a still condition the initial transformation is set to the 'Identity affine transformation' meaning that nothing is changed. As mentioned in [17] an affine transformation is a linear mapping method that preserves points, straight lines and planes.

- *Point Cloud Acquisition*: this block it's simply the acquisition of the frame, as a point cloud, that is going to be processed.

- *Transformation computation*: the core of the algorithm is inside this block which is further divided into different blocks

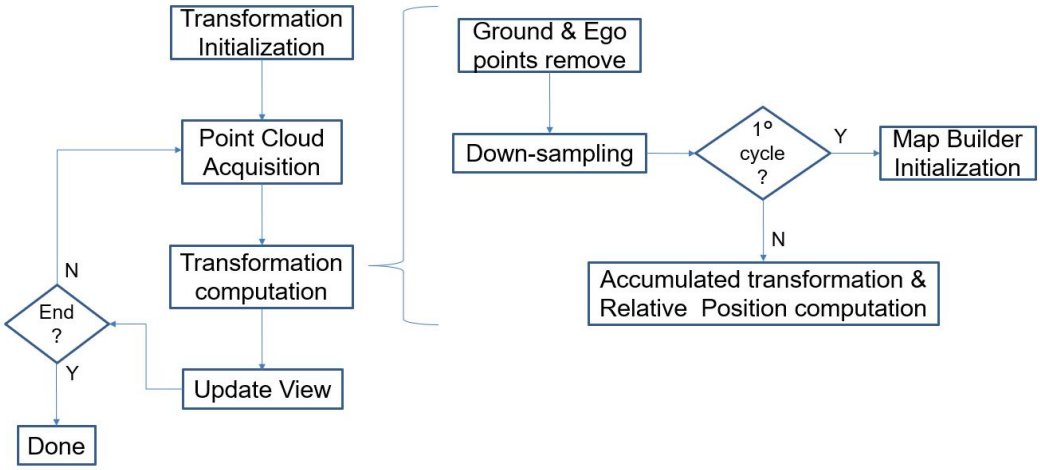


Figure 2.1: *Flowchart of the Localization and mapping algorithm.*

  - *Ground & Ego points removal*: it's part of the initial filtering, first the points related to the ground and to the vehicle surrounding area are removed because they are not useful for the transformation estimation.
  - *Down-sampling*: also here there is a filtering that is done by down-sampling the point cloud coming from the first filtering.
  - *Map Builder Initialization*: this block it's a simple initialization that creates the first map that is going to be updated and the relative pose as the origin point.
  - *Accumulated transformation & Relative Position computation*: here is where the scan-matching phase is executed. Thanks to the ICP or NDT algorithm the transformation between the last processed frame and the actual frame is estimated and the accumulated transformation is then computed. With that the relative position is calculated.

- *Update view*: in this block the global map is updated with the current frame and the relative position.

So, for the first guess of the transformation an identity matrix is used:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Applying this transformation to an object nothing changes, then its position and orientation remain the same.
The acquired point cloud is a set of point in a 3D space that represents the external environment around the vehicle. An example of that is in Fig.2.2: the ground points are all those points that define the circles in the figure and the ego points are defined as all the points that are in the neighbor of the LiDAR sensor located within a certain radius.

The pre-processing phase, as mentioned before, consists in two consecutive filtering

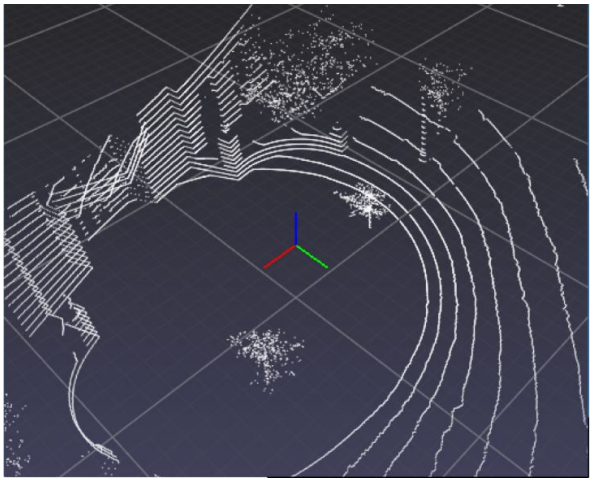


Figure 2.2: *Example of the original point cloud from an acquisition.*

steps that erase from the frame all the points out of our interest and It's done by using first a plane fitting function, for ground points, and a searching function, for the 'ego' points, and then using a random down-sampling method to further decrease the frame weight. Regarding the plane fitting method, a reference orientation and two maximum distances are defined: the idea is to take all the points that fit into a plane, within a maximum distance from it, that has no more than a specified angular distance from the reference orientation(Fig.2.3).

In Fig.2.3 is shown a 2D representation of the 3 ground removal parameters, but we have to keep in mind that It's like the plane that has to be fitted is rotating around the reference vector and continuously tilting between the two maximum positions.
For what concern the searching method there is only one parameter which is the radius within the search has to be done. This radius defines the space occupied by the vehicle.

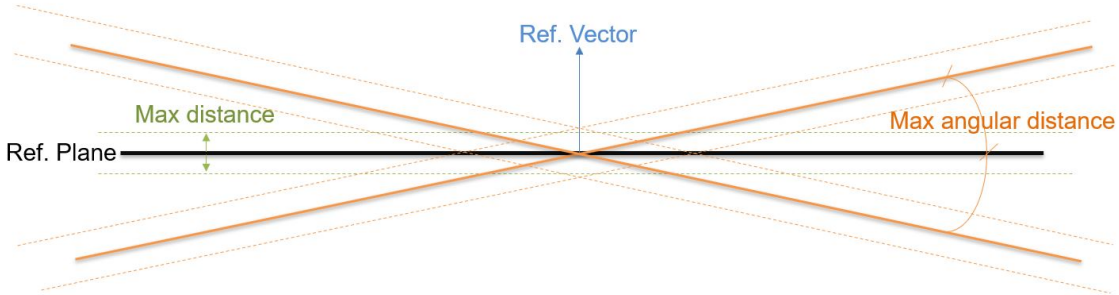


Figure 2.3: *Geometric definition of the plane fitting parameters .*

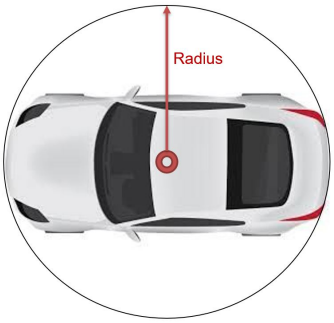


Figure 2.4: *Geometric definition of the searching radius parameter.*

In Fig.2.4 it's shown the geometric definition of this parameter.

The second and last step in pre-processing is then the down-sampling that has a unique parameter which is the portion of points we want to maintain in percentage, so if we want to keep the seventy percent of the frame the parameter has to be set to 0.7. Now our frame has the right format and it can be processed by the core of the algorithm, but first the global map has to be initialized: in the very first cycle the relative position is set to the origin point ([0,0,0] in our case), the map and the moving frames are set as the same point cloud.

Once the map is initialized here we are at the crucial phase of the entire process, where the scan-matching problem is addressed and solved thanks to the use of the ICP or NDT algorithm. These two relevant topics are going to be discussed in the next two sections of this chapter. At the end of this algorithm, there is the updating part where the overall map is overwritten with the last processed frame and the points of the relative position of the vehicle. An example of what we expect to have is shown in Fig.2.5.

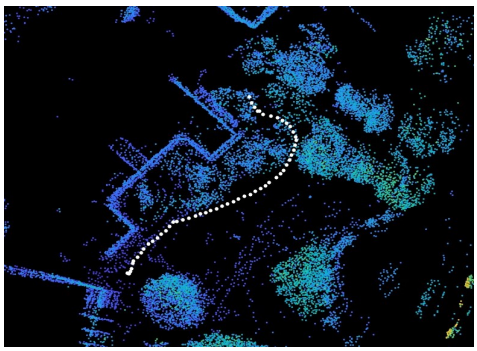


Figure 2.5: *Example of the map coming from the algorithm.*

2.2 Scan-matching problem

The scan-matching is the problem of registering two laser scans in order to determine the relative positions from which the scans were obtained, is one of the most heavily relied-upon tools for mobile robots. In our case the algorithm operates by performing a geometric match of the two scans and returns the best fused map obtained by merging the two partial map.

In other words is the calculation of translation and rotation of a scan to maximize its overlap with a reference scan. A several scan-matching algorithm have been proposed over the last decades; they are different in computational effort and way of performing the match.

As in [18] the method integrates two scans, S_1 and S_2, into a final map $S_{1,2}$. It is composed of three major steps: (1) determine the possible transformation of S_2 on S_1; (2) evaluate the transformations to identify the best transformation t of S_2 on S_1; (3) apply the best transformation to S_2, obtaining ${S_2}^t$, and fuse the segments of S_1 and of ${S_2}^t$ to obtain $S_{1,2}$.

This problem of matching it's also described as a point set registration problem, that is the task of finding the best fitting alignment between two sets of point samples. As we can see in Fig.2.6 the two scans have a simple rotation between them that once is computed it can be used to align the scans.

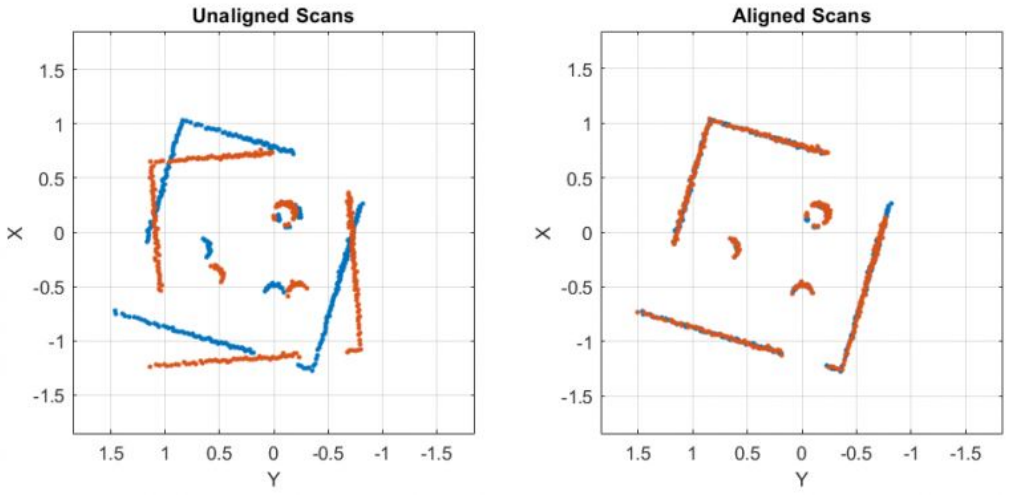


Figure 2.6: *Example of two matched scans.*

In this thesis work two different algorithm are proposed for the scan-matching problem, one is the Normal Distribution Transform algorithm(NDT) and the other is the Iterative Closest Point algorithm(ICP). Both of them will be discussed in the next two subsections.

2.2.1 Normal Distribution Transform

The NDT algorithm is a registration method that uses standard optimization techniques applied to statistical models of 3D points to determine the most probable registration between two point clouds. This method was introduced in [19] where the idea of matching is to have the maximum value from the score of the moving frame obtained through the alignment of this points on the probability density built from the reference scan. So the normal distribution is the one built from the reference scan(or target scan) and it's a piece-wise continuous and differentiable probability density. First a cell grid is created then for each cell, that contain at least 3 points, the mean and covariance parameters are computed:

- $q = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the mean
- $\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - q)(x_i - q)^t$ is the covariance

The creation of the grid introduce a discretization problem that in first place was solved by creating 4 target grids where each grid is translated by half of the cell size in single direction, then as in [20] a multi-layer approach was implemented to get more efficiency. Then the probability of measuring a sample at 2D-point x contained in this cell is modeled as a normal distribution N(q,Σ) and it is: $p(x) \approx exp(-\frac{(x-q)^t\Sigma^{-1}(x-q)}{2})$. So once the probability "grid" is constructed the definition of the transformation is needed. The mapping T between two coordinate frames is given by a rotation and a translation:

$$\mathbf{T}: \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{2.1}$$

So the final equation could be: $x' = \mathbf{R}x + \mathbf{t}$ where $\mathbf{R}$ represent the rotation, $\mathbf{t}$ represent

the translation and both describe the transformation **T**.
Next there is the scan alignment process where the goal is to obtain these two parameters using the laser scans taken at two positions.
The main steps of this approach, given two different scans(for example the first two) are the following:

1. Build the NDT of the first scan
2. Parameters initialization: with zero or using some odometry data
3. Map the reconstructed 2D point into the coordinate frame of the first scan(target scan) according to the parameters for each sample of the second scan(source scan)
4. Compute the NDT for each mapped point
5. Determine the score: done by evaluating the distribution for each mapped point and summing the result
6. By means of Newton's algorithm, optimize the score to calculate a new parameter estimate
7. Go to 3 until a convergence criterion is met

Mapping the second scan is done using T and could be considered optimal if the sum evaluating the normal distribution of all points, with the mean and the covariance parameters, is a maximum. This is going to be called the *score* of p, where p is the vector containing the parameters of rotation and translation.

$$\mathbf{score}(p) = \sum_{i=1}^{n} exp(-\frac{1}{2}((T(x_i, p) - q_i)^T \Sigma^{-1}(T(x_i, p) - q_i))) \tag{2.2}$$

The optimization of the *score* is done as a minimization problem, so the Newton's algorithm iteratively finds the parameters p that minimize a function f(that is equal to -*score*) by itertively solving the equation:

$$H\Delta = -g \tag{2.3}$$

where g is the transposed gradient of f and H is the hessian of f. The solution of this linear system is an increment Δp which is added to the current estimate. If H is positive definite, f(p) will initially decrease in the direction of Δp, if this is not the case, H is replaced by H'= H + λI, with λ chosen such that H' is safely positive definite. Some practical details about the minimization process can be found in [21].
Once the transformation has been estimated and optimized, a position tracking stage is performed in order to track the current position from a given time t. This is performed with respect to the target scan(*keyframe*), thus for the time t_k, the algorithm performs the following steps:

1. Define δ to be the estimate of the movement between time t_{k-1} and t_k
2. Then the position estimate of time t_{k-1} is mapped according to δ
3. So the optimization algorithm is executed using: the current scan, the NDT of the target and the new position estimate
4. Finally check if the convergence criterion is met, so whether the *keyframe* is 'near' enough to the source scan. If yes iterate, otherwise take the last successfully matched scan as a new *keyframe*

The 'near' enough condition is dictated by a simple empiric criterion involving the translation and the angular distance between the target and the source scan and the resulting score.
Finally the map is built using the angles θ_i and the translation vectors t_i that describe the pose of each scan i in the global coordinate frame. From the transformations, the map is overlapped for each frame and then optimized to reach a final goal where there is the global map with inside the trajectory way-points like in Fig.2.7.

2.2.2 Iterative Closest Point

The iterative closest point algorithm is another important registration method that was first introduced in [22] and now continue to be heavily used in point-set registration. Roughly speaking the ICP iteratively tunes the position of two point clouds through the optimization of square distances between corresponding pair of points from two

Figure 2.7: *Example of the SLAM mapping resulting from the algorithm.*

clouds. The approach that is going to be presented is the one explained in [10].
The method has brilliant convergence property in terms of local-minimum of a mean-square distance metric, so starting from an initial guess of translations and rotations the minimization is globally guaranteed and this is the reason why ICP is strongly used in motion estimation between scans where the correlation is not known.
To have a complete description of the method a few preliminary definitions have to be stated.

Definitions

For the transformation definition a rotation and a translation vectors have to be defined: in the case of rotation a quaternion vector is used, $\overrightarrow{q_R} = [q_0\ q_1\ q_2\ q_3]^t$, instead for translation a simple vector is used, $\overrightarrow{q_T} = [q_4\ q_5\ q_6]^t$, then the complete transformation(registration) vector is $\overrightarrow{q} = [\overrightarrow{q_R} \mid \overrightarrow{q_T}]^t$.
To estimate this two vectors some further definitions are required, for example: the *scans* to be processed, the *mean square objective function*, the *cross-covariance matrix*, the *distance metric*. From all of this entities two operator are defined: the least square quaternion operator **LSQ** that returns the *transformation quaternion* and the *mean square point matching error*, and the closest point operator **CP** that returns the set of closest point to a given point-set.
The two different point-sets, that are going to be called *scans*, are: the moving scan(or measured scan) represented by the point-set $M = \{\overrightarrow{m_i}\}$ which is the one to be aligned with the fixed scan(or target scan) that is represented by the point-set $X = \{\overrightarrow{x_i}\}$. N_x

and N_m are the number of points of each scan and $N_x = N_m$.
The two centroids, $\overrightarrow{c_m}$ for the measured scan M and $\overrightarrow{c_x}$ for the target scan X are given by:

$$\overrightarrow{c_m} = \frac{1}{N_M}\sum_{i=1}^{N_P}\overrightarrow{m_i} \;\; and \;\; \overrightarrow{c_x} = \frac{1}{N_X}\sum_{i=1}^{N_X}\overrightarrow{x_i} \tag{2.4}$$

and they are used for the definition of the *cross-covariance matrix* K_{mx} of the scans M and X

$$\mathbf{K_{mx}} = \frac{1}{N_M}\sum_{i=1}^{N_M}[(\overrightarrow{m_i} - \overrightarrow{c_m})(\overrightarrow{x_i} - \overrightarrow{c_x})^t] = \frac{1}{N_M}\sum_{i=1}^{N_M}[\overrightarrow{m_i}\overrightarrow{x_i}^t] - \overrightarrow{c_m}\overrightarrow{c_x}^t \tag{2.5}$$

that, in turn, is used to define the matrix from which $\overrightarrow{q}$ is obtained. This matrix is a composition of a column vector, $\Delta = [A_{23}\ A_{31}\ A_{12}]^T$, made from the entries of $A = (K_{mx} - K_{mx}^t)_{ij}$. So the 4x4 matrix $Q(K_{mx})$ is formed

$$\mathbf{Q(K_{mx})} = \begin{bmatrix} tr(K_{mx}) & \Delta^T \\ \Delta & K_{mx} + K_{mx}^T - tr(K_{mx})I_3 \end{bmatrix} \tag{2.6}$$

where I_3 is the 3x3 identity matrix and $tr(K_{mx})$ is the trace of the cross-covariance matrix. The unit eigenvector $\overrightarrow{q_R} = [q_0, q_1, q_2, q_3]^t$ corresponding to the maximum eigenvalue of the matrix $Q(K_{mx})$ is selected as the optimal rotation and is given by:

$$\overrightarrow{q_T} = \overrightarrow{c_x} - R(\overrightarrow{q_R})\overrightarrow{c_p} \tag{2.7}$$

The estimate of the mean square point matching error is obtained through the minimization of the *mean square objective* function, MSE:

$$MSE(\overrightarrow{q}) = \frac{1}{N_M}\sum_{i=1}^{N_M}||\overrightarrow{x_i} - R(\overrightarrow{q_R})\overrightarrow{m_i} - \overrightarrow{q_T}||^2 \tag{2.8}$$

Finally the Least Square Quaternion operator is denoted as:

$$(\overrightarrow{q}, d_{ms}) = LSQ(M, X) \tag{2.9}$$

where d_{ms} is the mean square point matching error. The notation $\overrightarrow{q}(M)$ is used to indicate the point-set M after transformation by the registration vector $\overrightarrow{q}$.
The other operator requires the definition of the distance metric between a single data point $\overrightarrow{m}$ and the target scan X which is

$$\mathbf{d}(\overrightarrow{m}, \mathbf{X}) = \min_{\overrightarrow{x} \in X} ||\overrightarrow{x} - \overrightarrow{m}|| \tag{2.10}$$

that will give the minimum distance denoted as $\overrightarrow{y}$. This vector $\overrightarrow{y}$ is such that $d(\overrightarrow{m}, \overrightarrow{y}) = d(\overrightarrow{m}, X)$ where $\overrightarrow{y} \in X$. Then the Closest Point operator is denoted as:

$$\mathbf{Y} = \mathbf{CP}(M, X) \tag{2.11}$$

where Y is the closest point set coming from the **CP** operator.

Algorithm

Now that the required definitions have been stated the description of the algorithm will be done in the following. As already outlined, a measured shape M is moved(source scan) "toward" a target shape X in order to obtain the optimal alignment capable to reduce as much as possible the matching error. In this work the data shape considered are always point-clouds.
In Fig.2.8 the general steps of the ICP algorithm are shown. The ICP algorithm can

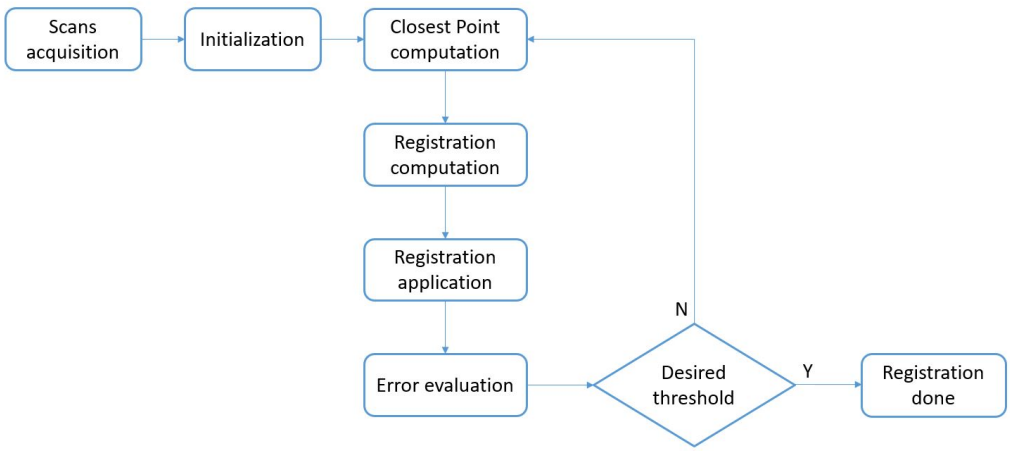


Figure 2.8: *General flowchart of the ICP algorithm.*

now be stated:

1. *Scans acquisition*: The two scans, M with N_M points $\overrightarrow{m_i}$ and X with N_X points $\overrightarrow{x_i}$, are given to the process.

2. *Initialization*: Here the registration vector and the first moving scan are set, in particular $M_0 = M$ and $\overrightarrow{q_0} = [1\,0\,0\,0\,0\,0\,0]^t$. Starting from $k = 0$ at each iteration the transformation is accumulated relative to the initial M_0.

3. *Closest Point computation*: in this block the **CP** operator is applied as $Y_k = \mathbf{CP}(M_k, X)$ and the first estimate of the closest point-set Y_k is obtained

4. *Registration Computation*: once the closest point is computed the **LSQ** operator is applied as $(\overrightarrow{q_k}, d_k) = \mathbf{LSQ}(M_0, Y_k)$ and a first estimate of the registration vector $\overrightarrow{q_k}$ is computed with its relative mean-square error d_k

5. *Registration application*: then all the points belonging to the moving scan are updated using the registration vector as $M_{k+1} = \overrightarrow{q_k}(M_0)$

6. *Error evaluation*: if the change in term of mean-square error is below a certain threshold $(d_k - d_{k+1} < \tau)$ the iteration is terminated otherwise the flow come back to step (3)

7. *Registration done*: if the desired threshold is reached the registration estimation is done and the transformation vector is returned as the good one

The expected result is the same as for the NDT algorithm, as shown is Fig.2.7, so a map composed by the points relative to the estimated path of the vehicle and the points corresponding to the environment.

2.3 Boundaries construction

In this section the boundaries construction approach that I've implemented is presented. The Fig.2.9 shows the simple flowchart that is used to extract the boundaries around the vehicle. The main idea is similar to the one for the mapping procedure: a first part of pre-processing to reduce the payload and then the processing part where

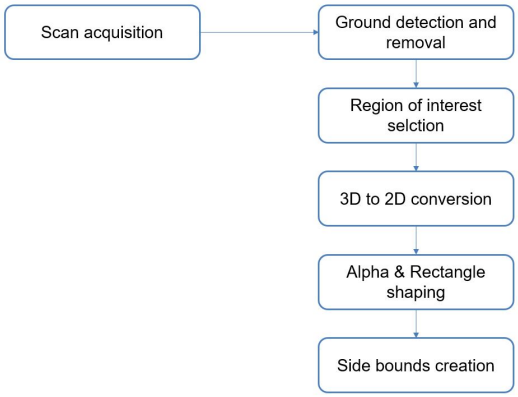


Figure 2.9: *General flowchart of the boundaries extraction algorithm.*

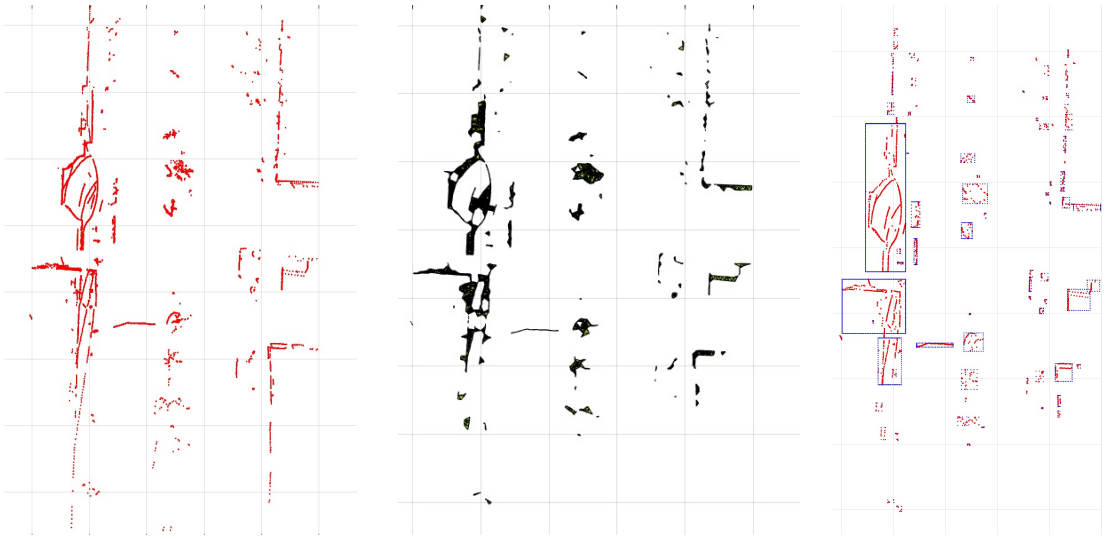
the bounds have been created.

All the assumption made in this working part are the result of some approximation that have been made working on sample data of a Velodyne VLP-16, the same used for the the testing phase, downloaded from the web. So all the results of this part are not very good but they can be used in future to develop the process of boundaries extraction. Referring to Fig.2.9 a brief description of each block is in the following:

- *Scan acquisition*: It's a simple acquisition of the scan data that from now on are going to be called 'frame'
- *Ground detection and removal*: It's quite the same procedure made for the mapping algorithm, so the definitions of a reference direction(the z-axis direction pointing to the sky) and two maximum distances(linear and angular) are done in order to select the point belonging to the ground. Once selected they are removed from the frame
- *Region of interest selection*: a portion of the 3D space is selected to reduce the computation weight of the frame. The parameters for this part depend on where the sensor is located, for example if the sensor is on the roof of the vehicle at the middle point, the region of interest(ROI) will be different from the one used if the sensor is located to the front of the vehicle
- *3D to 2D conversion*: this is a simple compression of all the frame-points into the $z = 0$ plane. So the z coordinate is simply set to zero.

- *Alpha & Rectangle shaping*: in this phase all the recognized 'shapes' inside the 2D view of the frame are converted into rectangles: first the determination of perimeter points from shapes then the rectangles building
- *Side bounds creation*: this last part is the most important because it returns the parameters related to the right and left bounds inside which the car can freely move. Usually an interpolation function is used

For what concern the '*Alpha & Rectangle shaping*', the shapes are recognized based on a radius that represents a disk: all the points inside that disk are labeled as a shape. Once the shapes are labeled, for each one the perimeter points are extracted and then a rectangle associated to the shape is created. The Fig.2.10 represent the different



(a) *2D conversion result.* **(b)** *Alpha-shape result.* **(c)** *Rectangle-shape result.*

Figure 2.10: *From the 2D image to the rectangle shapes.*

stages of the frame:

(a) is obtained when the z coordinate is set to 0

(b) from the alpha shaping method

(c) from the rectangle shaping

In the *Side bounds creation* the RANSAC(random sample consensus) interpolation algorithm has been tested with poor results because not for all the frames the interpolation is done with the proper number of points, so sometimes the parameter-set of

the boundaries is resulting to be empty. A possible solution to overcome this problem is to create a fixed number of centroids to groups the side points.

Chapter 3

Validation

This chapter contain a brief description of the hardware devices composing the overall acquisition system used and a detailed description of the LiDAR sensor used, the Velodyne VLP-16.
Then different data-sets are defined based on the environment where the car was driven and tested varying the parameter-set of the algorithms. First a data-set downloaded from the support website of Velodyne has been used to test and validate the methods. The assumptions on the 'true' trajectory have been pointed out using a Velodyne file player that shows frame by frame the measured data of the sensor.
Then our experimental data acquired using our sensor have been tested.
This validation part is structured as follow: a first division based on the used algorithm, then for each data-set different parameter-set are used to see the principal differences. Indoor and Outdoor data-sets are the experimental ones containing data coming from our sensor.

3.1 Acquisition system

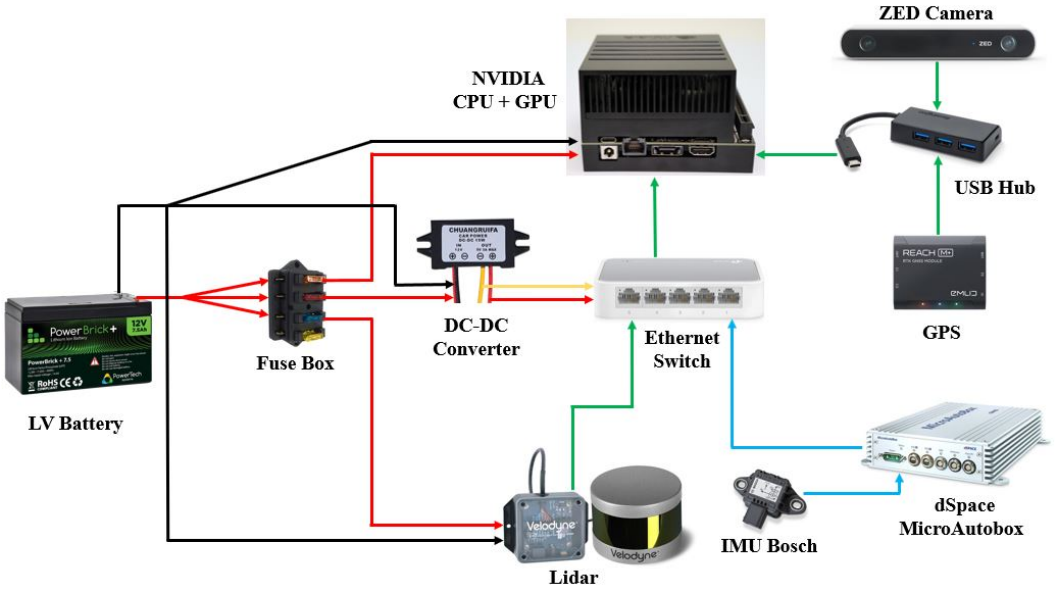


Figure 3.1: *General scheme of the overall acquisition system.*

As shown in Fig.3.1 the acquisition system is not so simple because it has a lot of sensors like camera, radar, GPS and IMU. The GPS is inside but it's not used. The power supply is a 12V Lithium-Ion battery, the NVIDIA graphic unit and the LiDAR interface are fed at 12V, while the ethernet switch is fed at 5V using a DC/DC converter. For safety a fuse box between the battery and the devices is inserted.
The IMU device is from BOSCH, it's connected to the dSpace unit which is a real-time system for performing fast function prototyping and it's used to collect inertia data like orientation, acceleration and speed.
The ZED camera is a stereo camera that has a depth perception indoors and outdoors at up to 20 m at 100FPS.
The NVIDIA graphic unit has both the GPU and the CPU integrated, it's loaded with a LINUX operating system that manage and fuse all the data coming from the different sensors.
The LiDAR sensor is a Velodyne VLP-16, it has an interface unit that is connected to the ethernet switch which is the device used to collect the LiDAR data. More detail about this sensor are outline in the next sub-section.

3.1.1 Velodyne VLP-16

Figure 3.2: *Velodyne VLP-16 3D LiDAR sensor.*

The Velodyne VLP-16 is a powerful 3D LiDAR with a range of 100 m. It's mostly used in robotics, autonomous vehicle, terrestrial 3D mapping and many other applications. It supports 16 channels in the vertical field of view with a span of 30°, 15° up and 15° down, with an angular resolution of 2.0°. In the horizontal field of view it has a span of 360° with an angular resolution in the range of 0.1°-0.4°. It can work with a rotation rate from 5 Hz up to 20 Hz, typically the measurements are made at 10 Hz. It can creates from ≈ 300000 to ≈ 600000 points per second depending on the operating mode.

Typically it has a low power consumption, 8 W, and it can operates in a range of temperature from -10°C to +60°C.

Its internal laser is classified as 'Class 1' eye-safe for the international standard IEC 60825-1 on laser safety. The experimental data used for validate our methods have been acquired with this LiDAR mounted on the '*Squadra Corse*' race car of Politecnico di Torino. More precisely it has been mounted at the center of the front wing.

3.2 Data-sets

In this section all the used data-sets are defined and described.

3.2.1 Monterey-Highway data-set

This data-set is the one downloaded from the reference website of the Velodyne VLP-16. As already said for this data-set strong assumptions, based on the stream of the file, were made.
The name of the data-set is related to the road where the data have been collected, so 'Monterey Highway' stands for the road on which the car has been driven to collect the data. From the streaming it can be seen that the car starts from a parking lot and it goes straight until a road crossing where it changes, with a 'U' turn, its direction coming back in the same road, then at another road crossing it does the same, so another 'U' turn.

3.2.2 Indoor data-sets

These data-sets contain the data that have been collected inside a building where small straight lines and turns have been done using some cones as delimiters and the car of 'Squadra Corse' has been pushed during all the measurements. Obviously the 3 different and relevant data-sets used are the streams of a straight line, a left turn and a right turn.

3.2.3 Outdoor data-sets

These data-sets contain the data that have been collected on a circuit. Also here the car has been pushed on straight lines and turns but longer than the Indoor data-sets. Data have also been measured with engine running, in particular the car have been driven in a straight road and in an little oval circuit: in both straight and oval the car was driven with low and high speed. The tests have been done for the data-sets where the engine is running.

3.3 Parameters sets

In this section the different parameters are explained in detail.
As already said in in Section2.1 there are 5 parameters defining the pre-processing phase:

- *Reference orientation*: it will be always the same, the orientation pointing up in the z coordinate
- *Max distance*: It's part of the ground-removal parameters and it will be called 'maxD', changing this distance the portion of points that will be remove from the frame is different. It depends also on where the sensor is located, so there will be a different value from the Monterey-Highway to the other data-sets
- *Max angular distance*: It's part of the ground-removal parameters and it will be called 'maxAngD'
- *Radius*: It represents the ego points removal and it will be called 'rad'. Different values from the Monterey-Highway to the other data-sets
- *Downsample Percent*: It will be called 'downS', changing this parameter the accuracy and payload will change

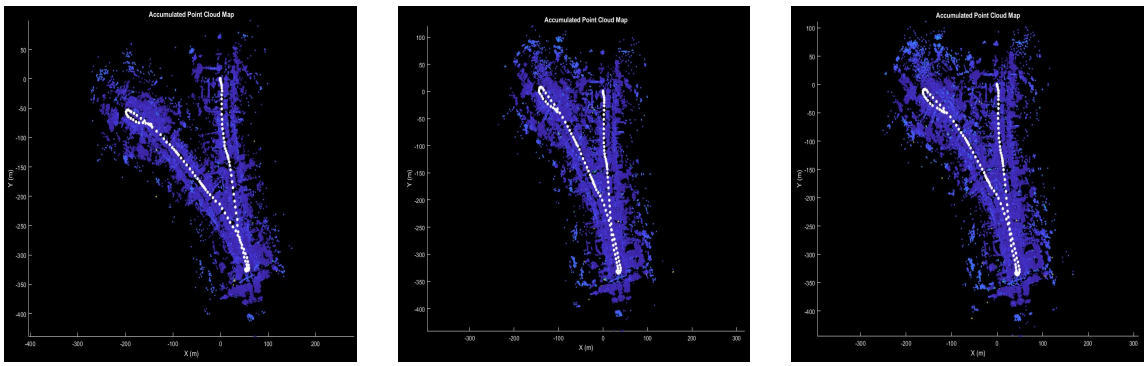
Then another parameter is defined as *skipFrames*('skipF' in the following): It defines the number of skipped frames that the algorithm will not use to compute the transformation, so increasing this value fewer frames per second will be used, on the contrary, decreasing this value more frames per second will be used. So this parameter determines the number of 'fps' the algorithm is going to use.
In some cases also a 'Region-Of-Interest' (ROI) will be defined through other two parameters, the distances in meters on the x and y coordinates, 'xL' and 'yL', that are two limit ranges: this ROI defines the portion of space selected in the frame before to process it(in the z coordinate any cut is done).

3.4 ICP tests

In this section all the tests that have been done using the ICP algorithm will be shown. As for the NDT algorithm tests the main approach is to change the parameters set in order to see the principal improvements or degradation of accuracy.

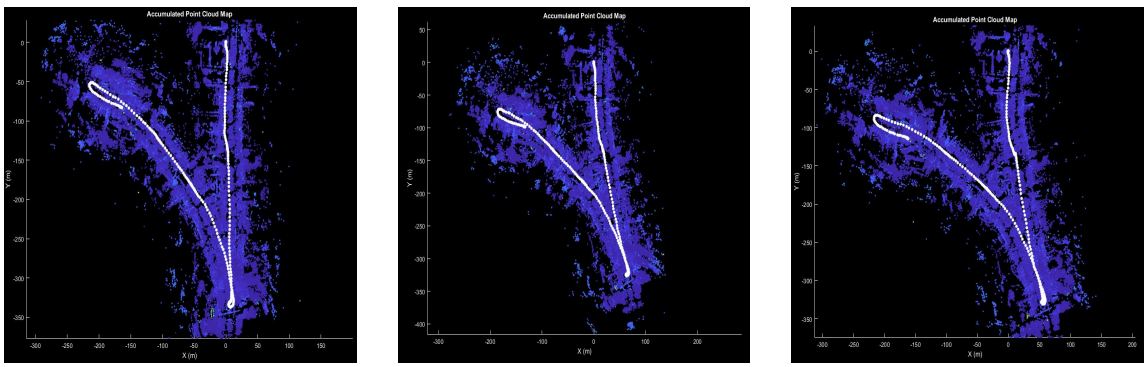
3.4.1 ICP on Monterey-highway data-set

First the ground removal parameters have been fixed, the parameter 'skipF' has been set to 10 to have 1 frame per second, while the 'downS' parameter has been changed from 0.1 to 0.7. Then a second test have been made fixing the parameters 'skipF' and 'downS' in order to have 2 frames per second and the minimum part of the entire frame, while the ground removal parameter were changing. Resulting simulations are shown in Fig.3.4 and Fig.3.3.



(a) *downS=0.1.* **(b)** *downS=0.3.* **(c)** *downS=0.7.*

Figure 3.3: *ICP tests in 'Monterey-Highway' for different down-sampling values.*



(a) *maxD=0.5.* **(b)** *maxD=0.55.* **(c)** *maxD=0.6.*

Figure 3.4: *ICP tests in 'Monterey-Highway' for different maximum angular distances.*

Additional tests were made with increased values of 'downS', but they are not shown due to their uselessness in this case. There will be comments about that in the next chapter.

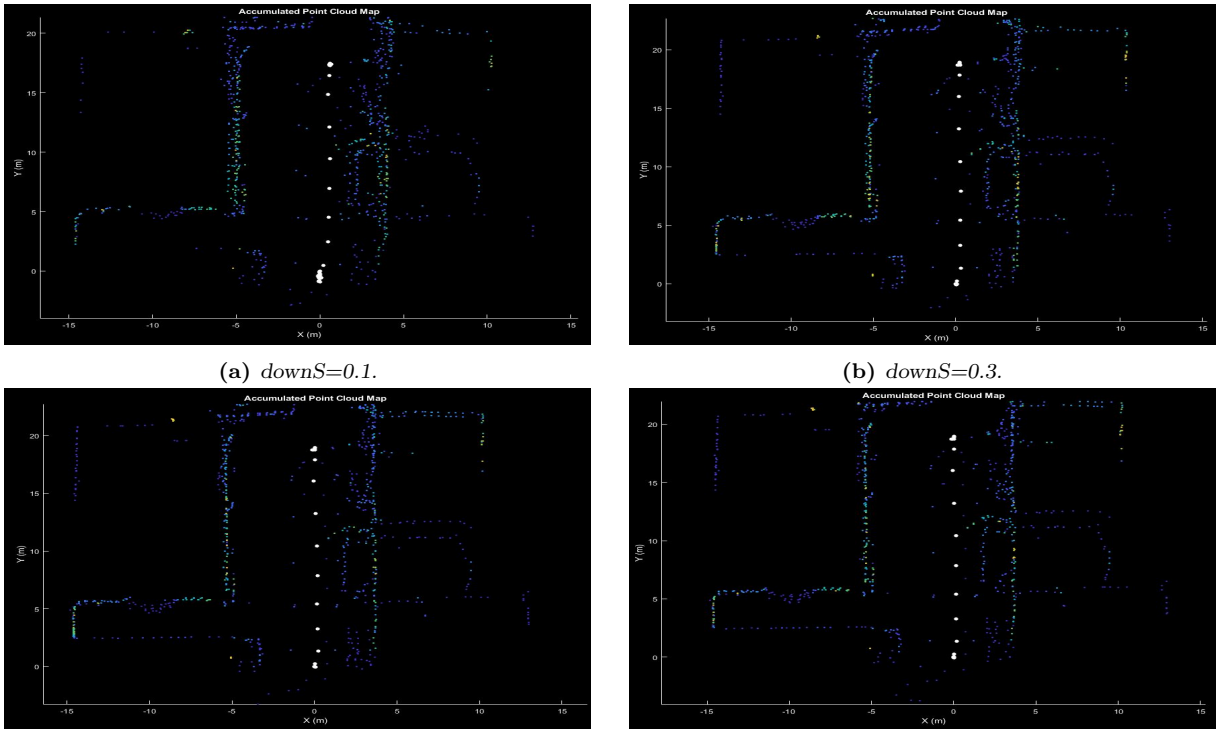
3.4.2 ICP on Indoor data-sets

Here there are 3 different cases: straight-line, left turn and right turn. In these tests the fixed parameters are:

- $rad = 1.2$
- $maxD = 0.08$
- $maxAngD = 10$

Straight-line

In the figure below, Fig.3.5, the main changes for different values of 'downS' are shown, keeping always 1 frame per second.



(a) *downS=0.1.*

(b) *downS=0.3.*

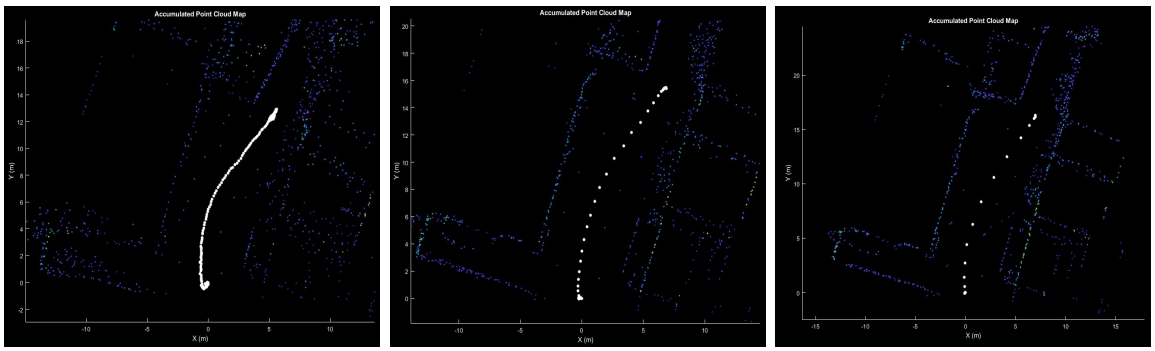
(c) *downS=0.7.*

(d) *downS=0.9.*

Figure 3.5: *ICP tests in an indoor straight line for different down-sampling values.*

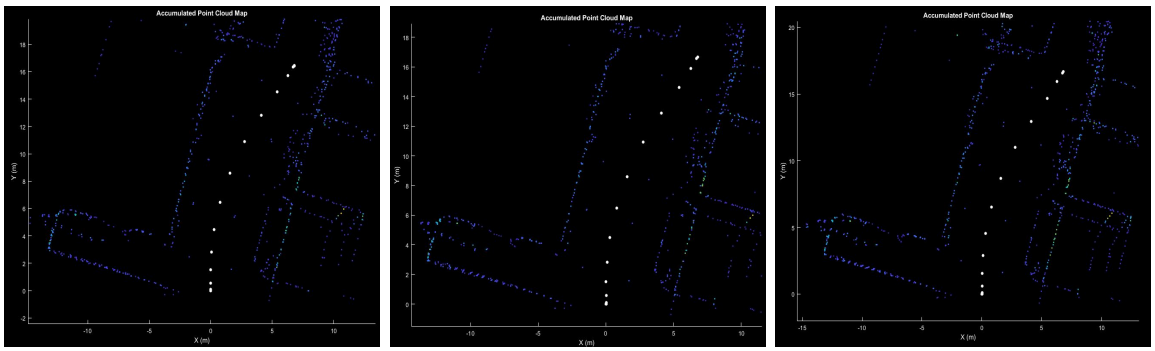
Right turn

In the figure below, Fig.3.6, the main changes for different values of 'skipF' are shown, keeping always 'downS' to 0.1. Then the number of frame per second has been set to 1 varying the downsampling as reported in Fig.3.7.



(a) *skipF=1.* **(b)** *skipF=5.* **(c)** *skipF=10.*

Figure 3.6: *ICP tests in an indoor right turn for different values of fps.*

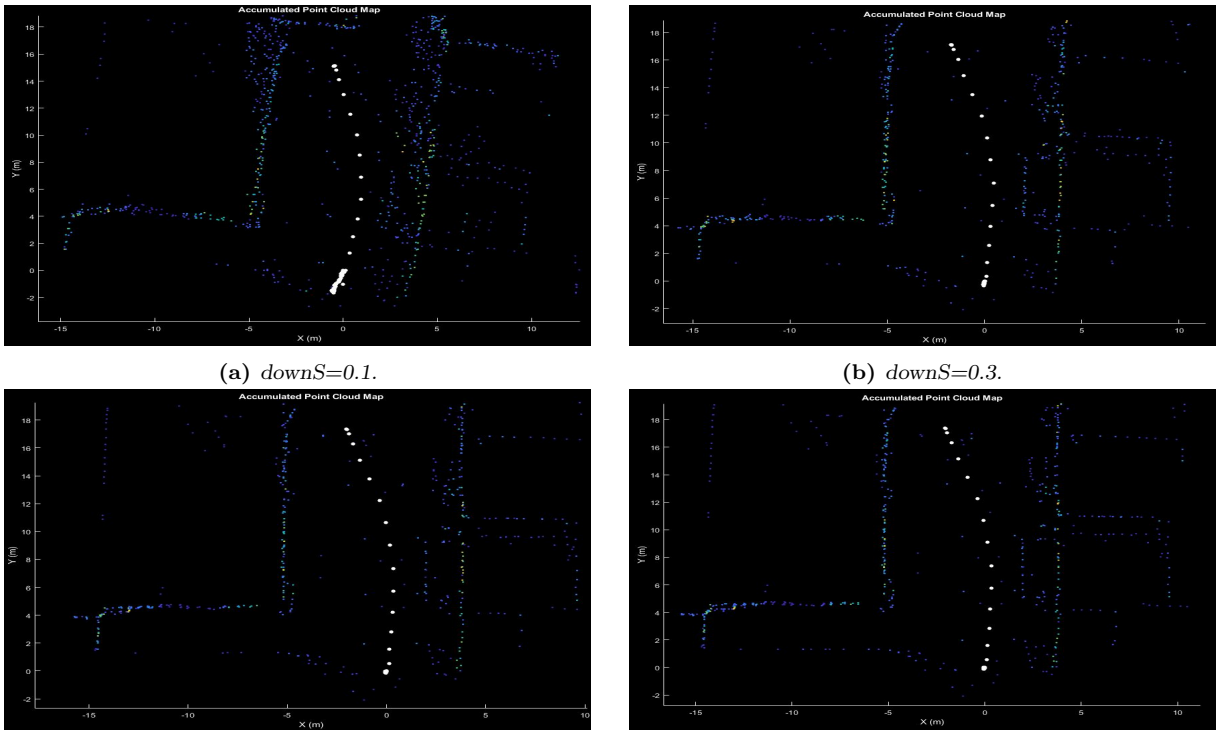


(a) *downS=0.3.* **(b)** *downS=0.7.* **(c)** *downS=0.9.*

Figure 3.7: *ICP tests in an indoor right turn for different down-sampling values.*

Left turn

In the figure below, Fig.3.8, the main changes for different values of 'downS' are shown, keeping always 1 frame per second.



(a) *downS=0.1.* **(b)** *downS=0.3.*

(c) *downS=0.7.* **(d)** *downS=0.9.*

Figure 3.8: *ICP tests in an indoor left turn for different down-sampling values.*

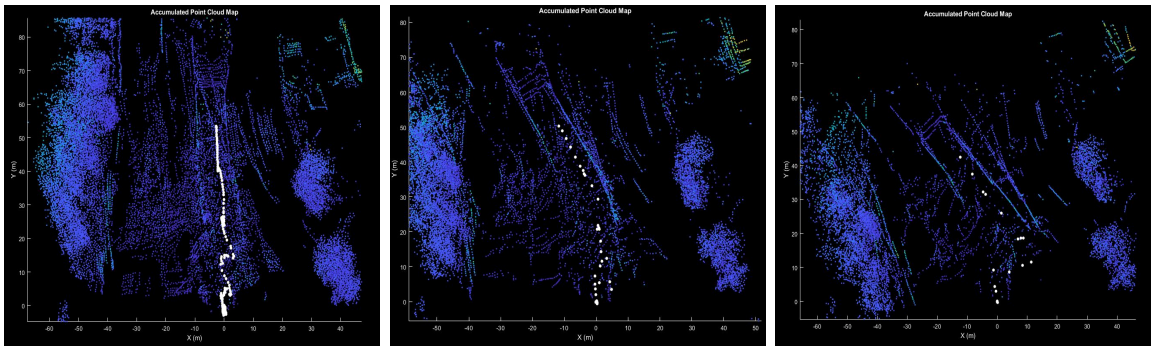
3.4.3 ICP on Outdoor data-sets

Here there are two different type of data-sets, 'Straight acceleration' and 'Ovale circuit', further divided into high speed set and low speed set.
For all this tests the following parameters remains constant:

- $rad = 1.2$
- $maxD = 0.08$
- $maxAngD = 10$

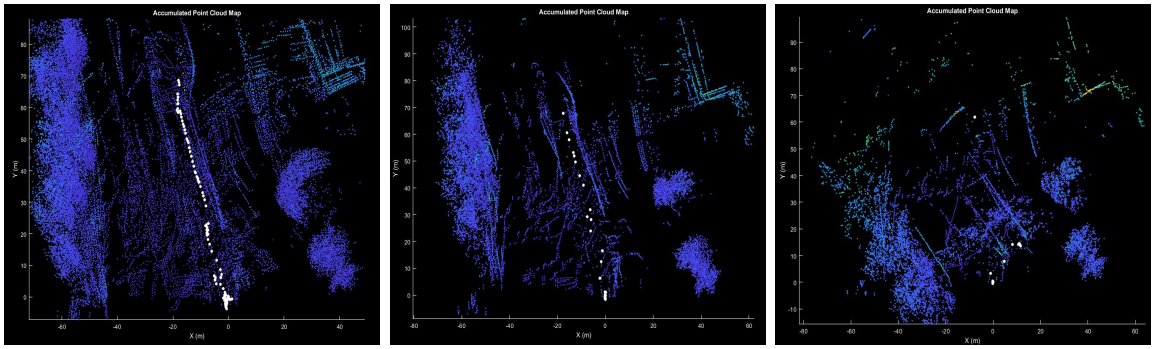
Straight acceleration

For both low and high speed sets, the 'downS' was set to 0.99, so almost all the points of the single frame have been used, while the number of frames per second changes. The differences are shown in Fig.3.9 for low speed and in Fig.3.10.



(a) *skipF=1.* **(b)** *skipF=5.* **(c)** *skipF=10.*

Figure 3.9: *ICP tests in an outdoor straight line for different values of fps at low speed.*

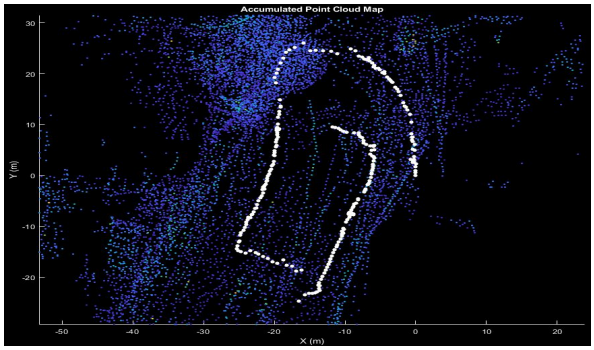


(a) *skipF=1.* **(b)** *skipF=5.* **(c)** *skipF=10.*

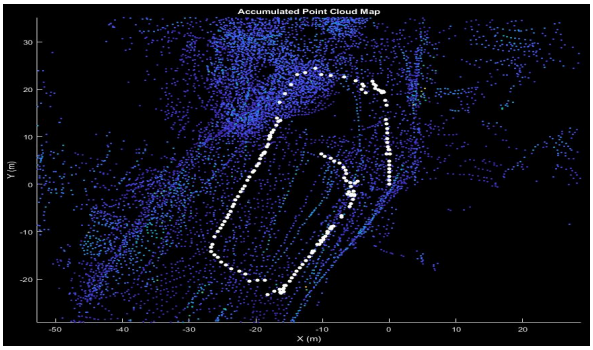
Figure 3.10: *ICP tests in an outdoor straight line for different values of fps at high speed.*

Ovale circuit

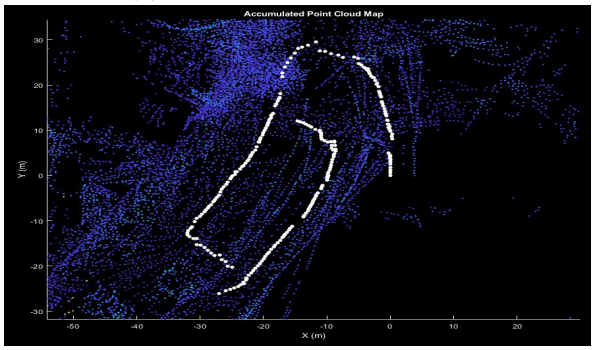
For this tests the simulation has been stopped at a predefined frame number corresponding to the first lap of the oval plus a little portion of circuit. In both low and high speed cases the number of frames per second was set first to 5 then to 3 and for both the down-sampling has been changed as in Fig.3.11.



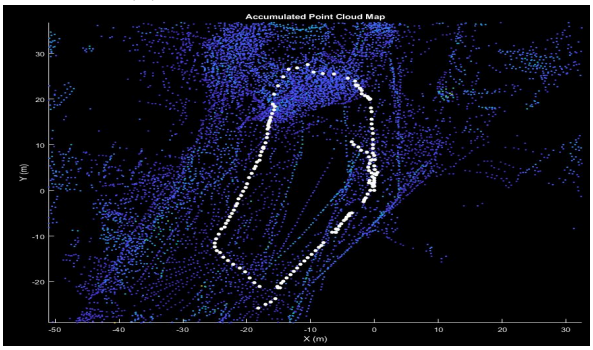
(a) *low speed, downS=0.5, fps=5.*



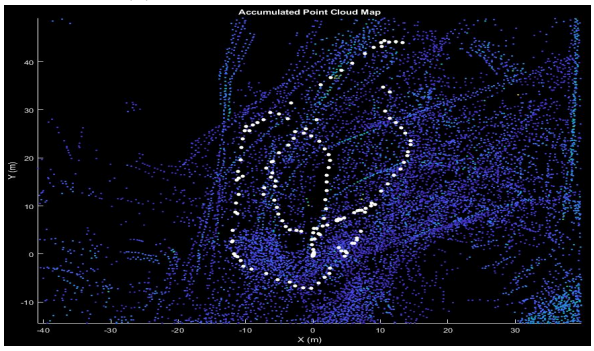
(b) *low speed, downS=0.5, fps=3.*



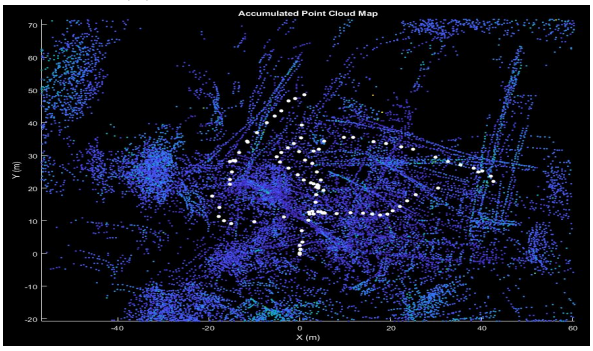
(c) *low speed, downS=0.95, fps=5.*



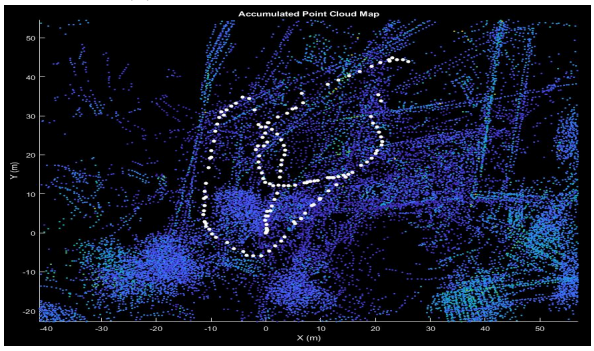
(d) *low speed, downS=0.95, fps=3.*



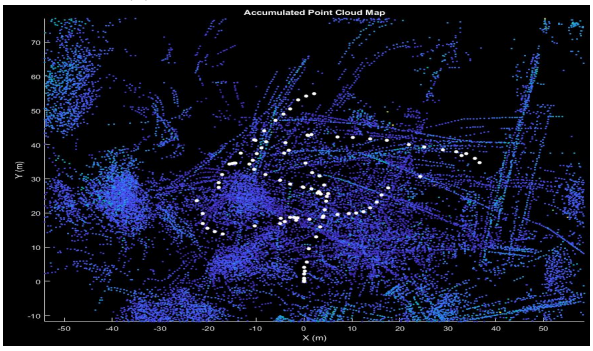
(e) *high speed, downS=0.5, fps=5.*



(f) *high speed, downS=0.5, fps=3.*



(g) *high speed, downS=0.95, fps=5.*



(h) *high speed, downS=0.95, fps=3.*

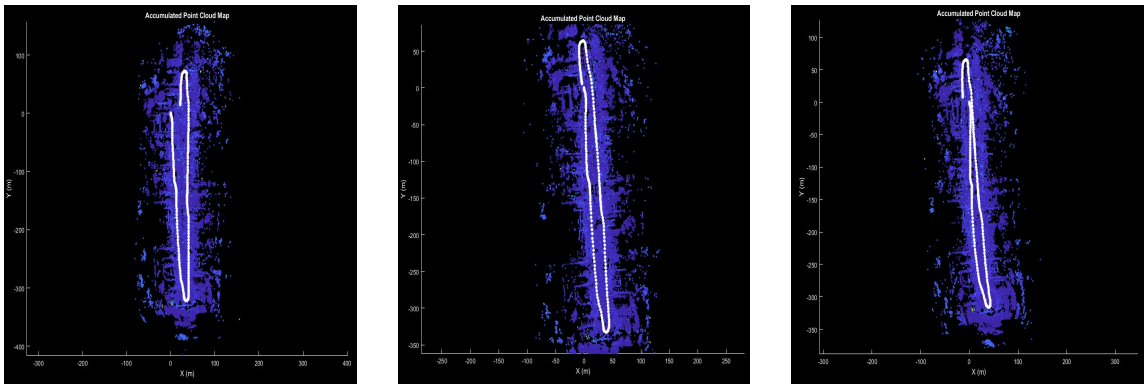
Figure 3.11: *ICP tests in an outdoor oval circuit for different down-sampling and number of fps values at low and high speed.*

3.5 NDT tests

In this section all the tests that have been done using the NDT algorithm will be shown.

3.5.1 NDT on Monterey-highway data-set

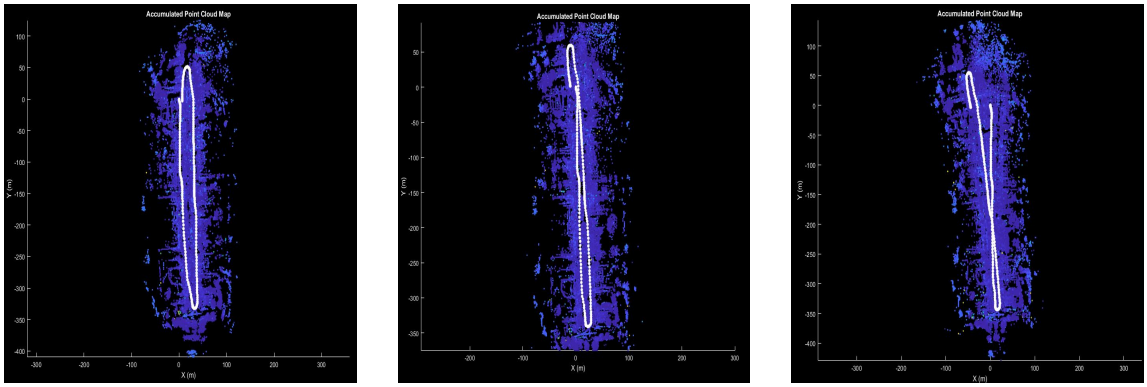
First the down-sampling parameter and the frame per second parameter have been fixed respectively to 0.1 and 2, while the ground removal parameter 'maxD' has been changed as shown in Fig.3.12.



(a) *maxD=0.5.* **(b)** *maxD=0.55.* **(c)** *maxD=0.5.*

Figure 3.12: *NDT tests in 'Monterey-Highway' for different maximum distances.*

Then a second test has been made using the 'ROI' defined by $xL = yL = 90$ and changing the down-sampling 'downS' parameter from 0.1 to 0.7(Fig.3.13).



(a) *downS=0.1.* **(b)** *downS=0.3.* **(c)** *downS=0.7.*

Figure 3.13: *NDT test in a 'ROI' of 'Monterey-Highway' for different down-sampling values.*

3.5.2 NDT on Indoor data-sets

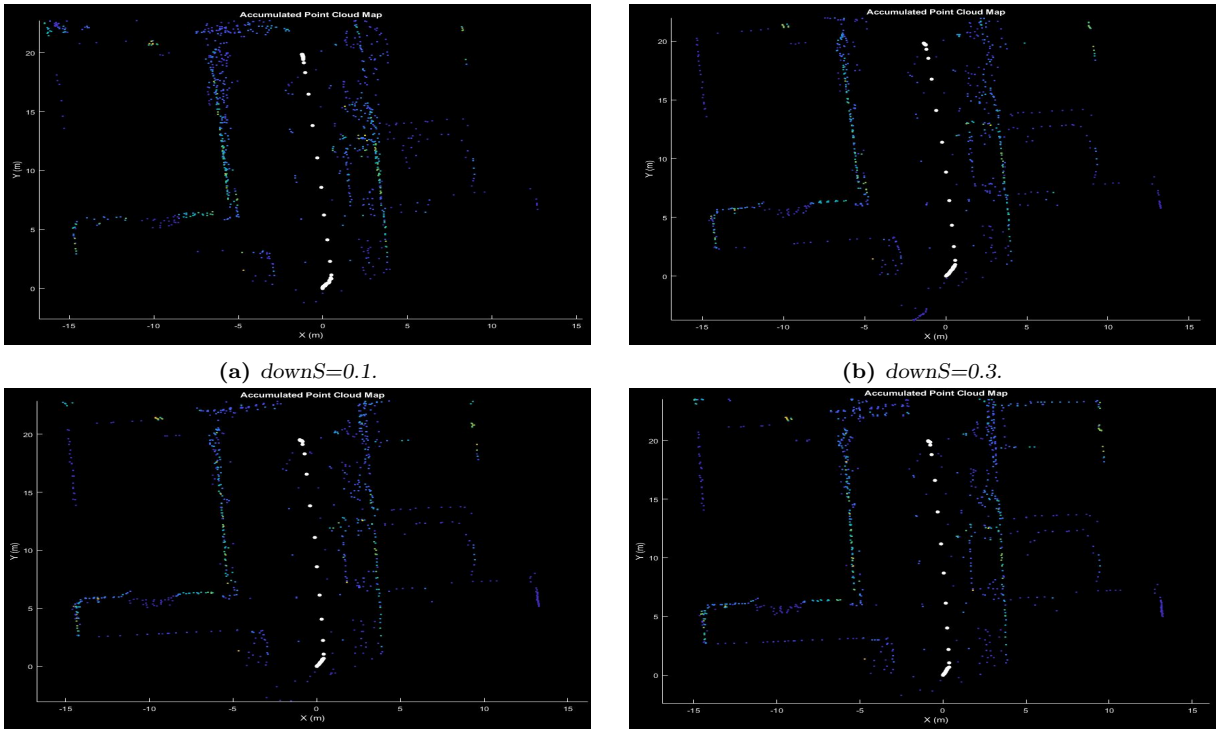
Here, as for the ICP tests, there are 3 cases: straight line, left turn and right turn. In these tests the fixed parameters are:

- $rad = 1.2$

- $maxD = 0.08$
- $maxAngD = 10$

Straight line

The frame per second parameter is maintained to 1 fps, while the downsampling parameter 'downS' ranges from 0.1 to 0.9(Fig.3.14).



(a) *downS=0.1.* **(b)** *downS=0.3.*

(c) *downS=0.7.* **(d)** *downS=0.9.*

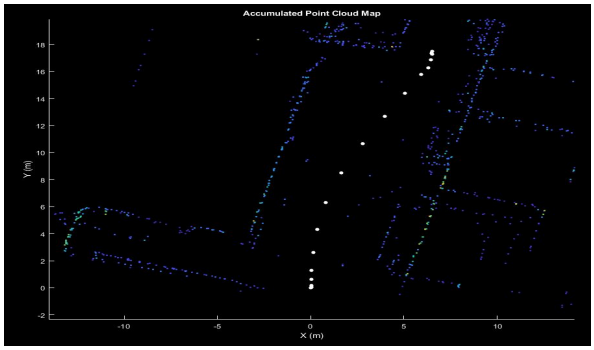
Figure 3.14: *NDT tests in an indoor straight line for different down-sampling values.*

Right and Left turn

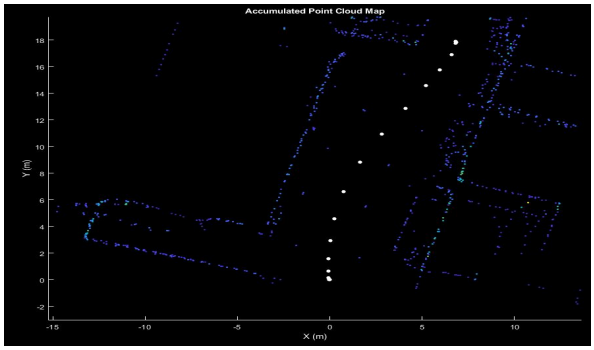
The same tests done for the straight line have been done also for the right and left turn data-sets, so a fixed fps to 1 and the parameter 'downS' ranging from 0.1 to 0.9(Fig.3.15 and Fig.3.16).

3.5.3 NDT on Outdoor data-sets

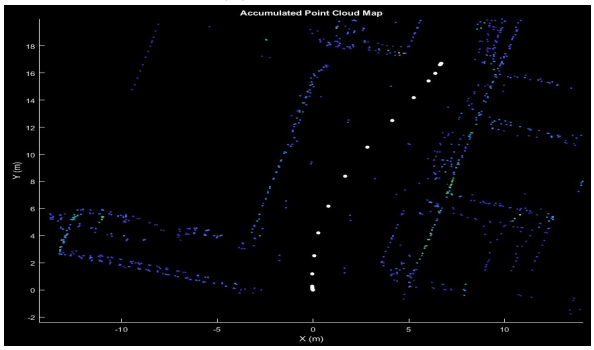
As already stated in the ICP tests here there are two different data-sets: 'Straight acceleration' and 'Ovale circuit', further divided into high and low speed. For all this



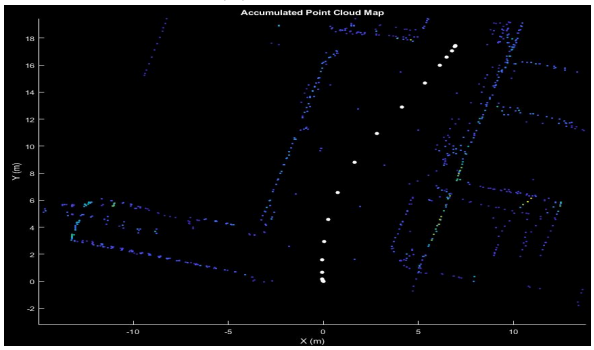
(a) *downS=0.1.*



(b) *downS=0.3.*

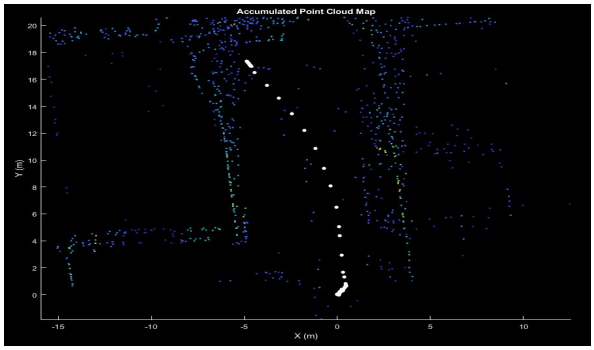


(c) *downS=0.7.*

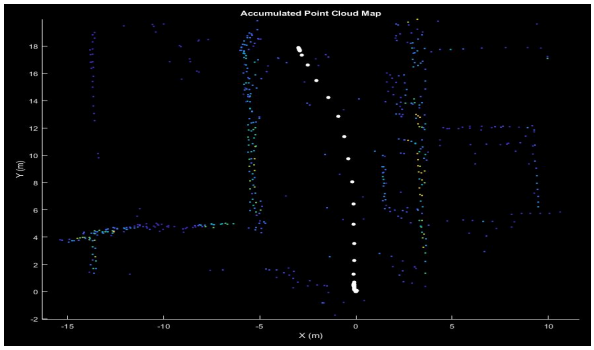


(d) *downS=0.9.*

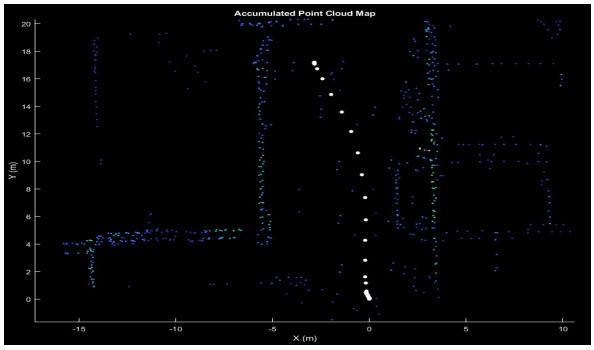
Figure 3.15: *NDT tests in an indoor right turn for different down-sampling values.*



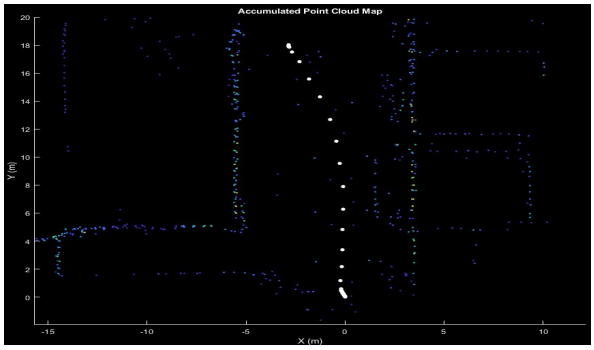
(a) *downS=0.1.*



(b) *downS=0.3.*



(c) *downS=0.7.*



(d) *downS=0.9.*

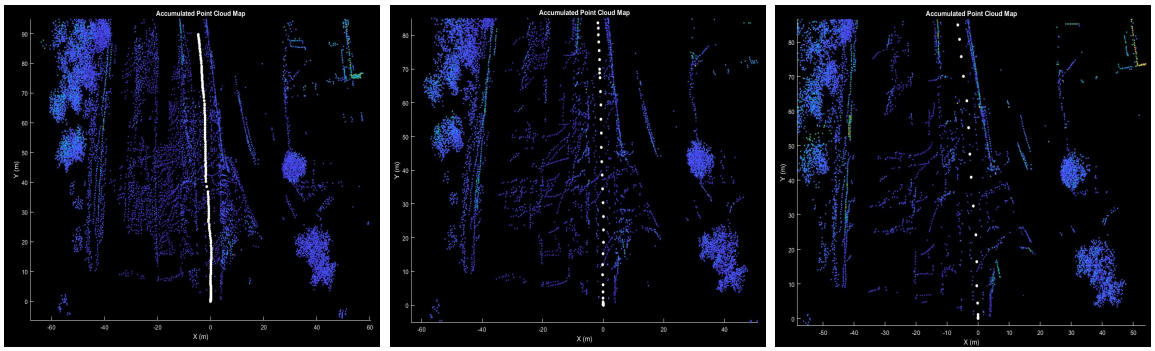
Figure 3.16: *NDT tests in an indoor left turn for different down-sampling values.*

tests the following parameters remains constant:

- $rad = 1.2$
- $maxD = 0.08$
- $maxAngD = 10$

Straight acceleration

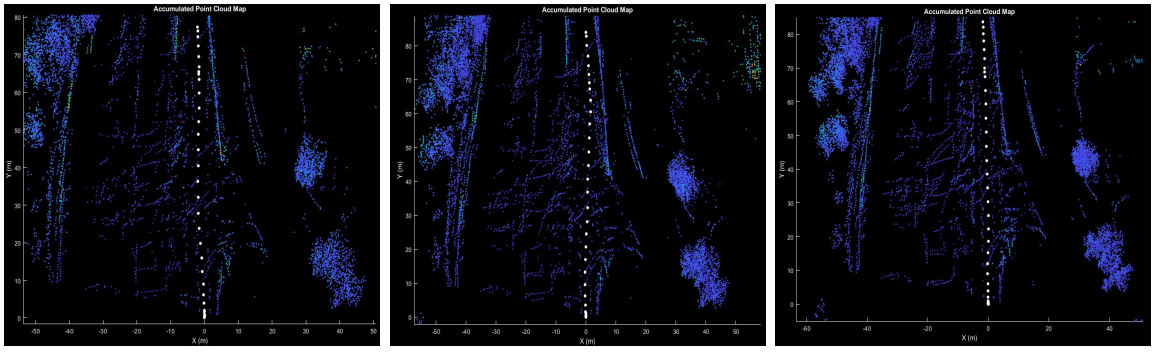
The first tests have been made changing the fps, so using almost all the points in the frame(downS=0.99) the parameter skipF has been changed from 1 to 10, as shown in Fig.3.17 that correspond to fps=10, fps=2 and fps=1.



(a) *skipF=1.* **(b)** *skipF=5.* **(c)** *skipF=10.*

Figure 3.17: *NDT tests in an outdoor straight line for different values of fps at low speed.*

Then the the varying parameter was the down-sampling that has been reduced from 0.99 to 0.3, keeping fps=2, as outlined in Fig.3.18.



(a) *downS=0.3.* **(b)** *downS=0.5.* **(c)** *downS=0.99.*

Figure 3.18: *NDT tests in an outdoor straight line for different down-sampling values at low speed.*

The last test was made setting the downsampling to 0.5 and decreasing the fps to

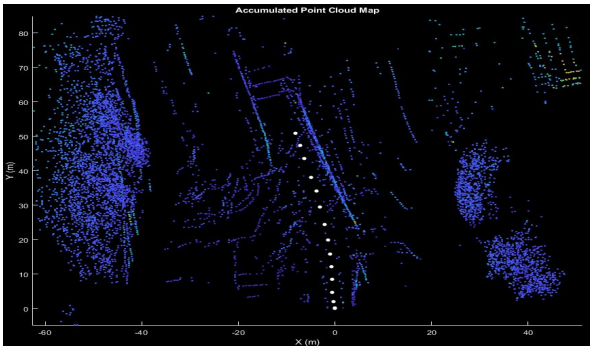
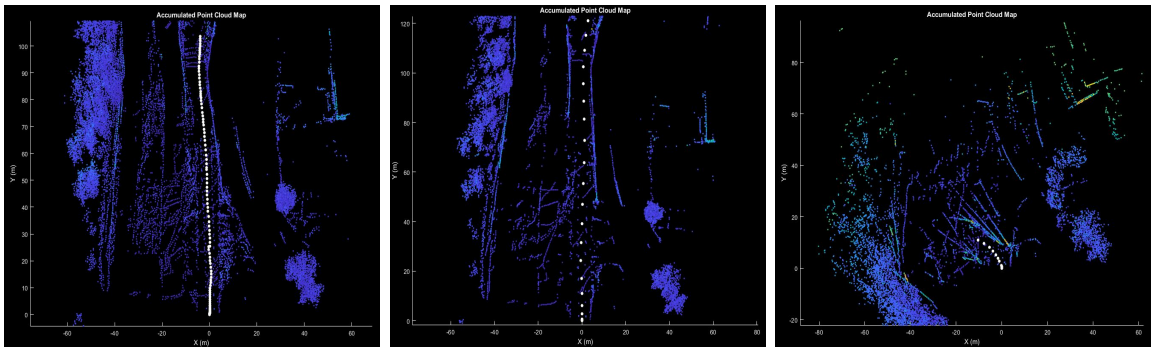


Figure 3.19: *NDT tests in Straight acceleration at low speed for fps=1 and downS=0.5.*

1(Fig.3.19):

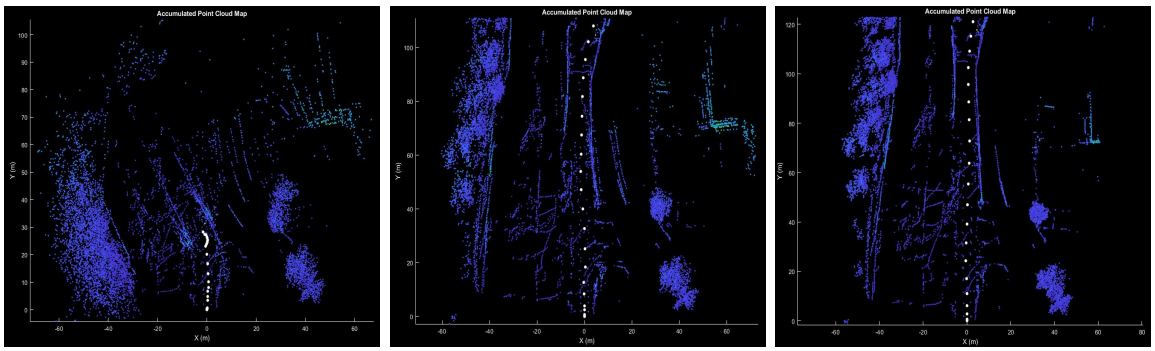
For what concern the high speed data-set, first the same tests as for low speed were made as shown in the following(Fig.3.20):



(a) *skipF=1.* **(b)** *skipF=5.* **(c)** *skipF=10.*

Figure 3.20: *NDT tests for 'Straight acceleration' at high speed for different fps.*

Then, keeping $fps = 2$, the downsampling has been changed in the range $0.5 - 0.99$ as illustrated in Fig.3.21

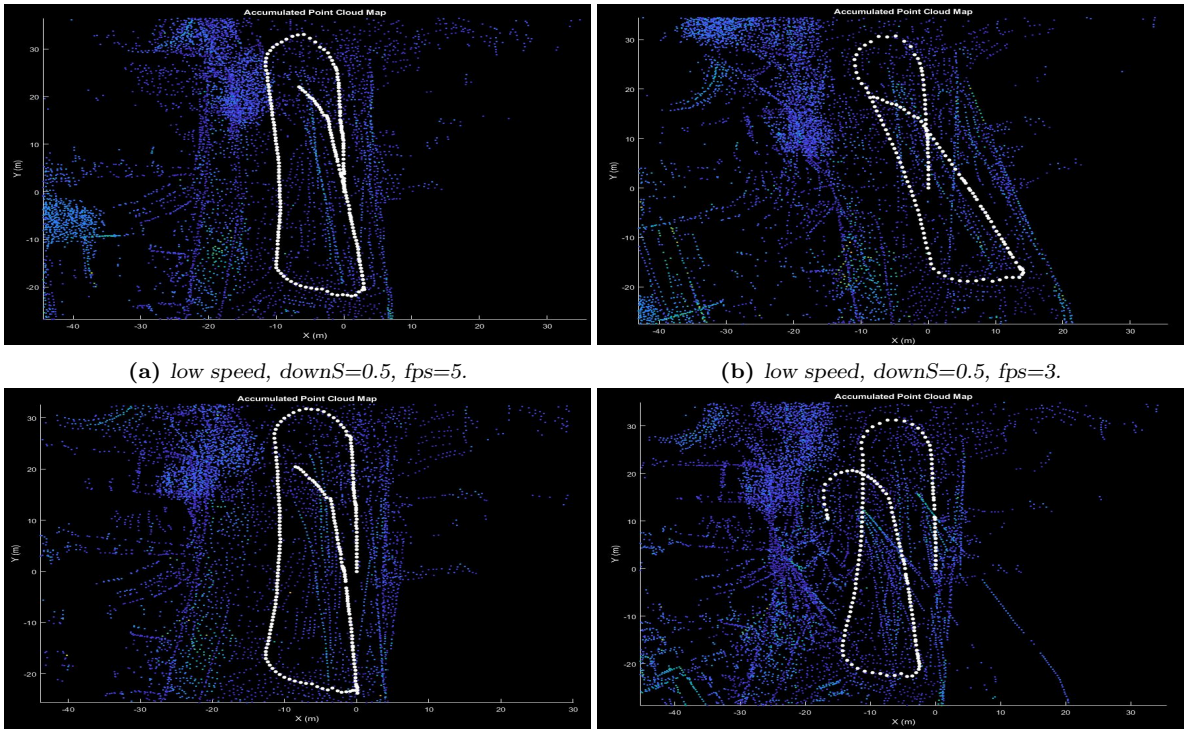


(a) *downS=0.5.* **(b)** *downS=0.7.* **(c)** *downS=0.99.*

Figure 3.21: *NDT tests for 'Straight acceleration' at high speed for different downsampling.*

Ovale circuit

The same tests done for the ICP algorithm have been made both at high and low speed, so $downS = 0.5, 0.95$, and $fps = 3, 5$. In Fig.3.22 the low speed results are shown. Then maintaining constant to 5 fps and 0.5 of downsampling, the 'maxD'



(a) *low speed, downS=0.5, fps=5.*

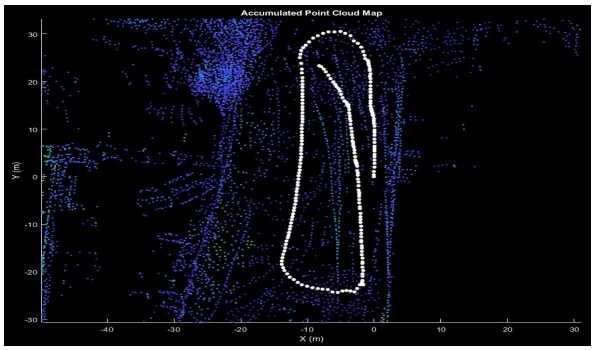
(b) *low speed, downS=0.5, fps=3.*

(c) *low speed, downS=0.95, fps=5.*

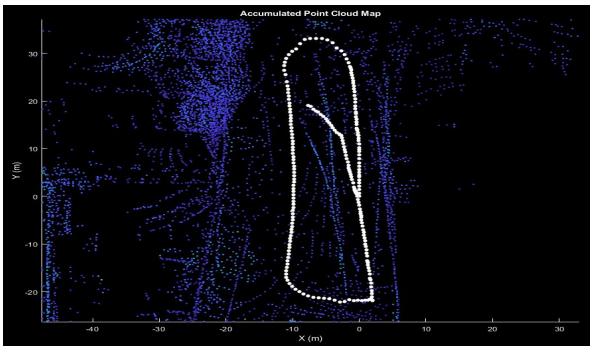
(d) *low speed, downS=0.95, fps=3.*

Figure 3.22: *NDT tests in oval circuit at low speed for different values of downsampling and fps.*

ground removal parameter has been changed as shown in Fig.3.23. A last test has been



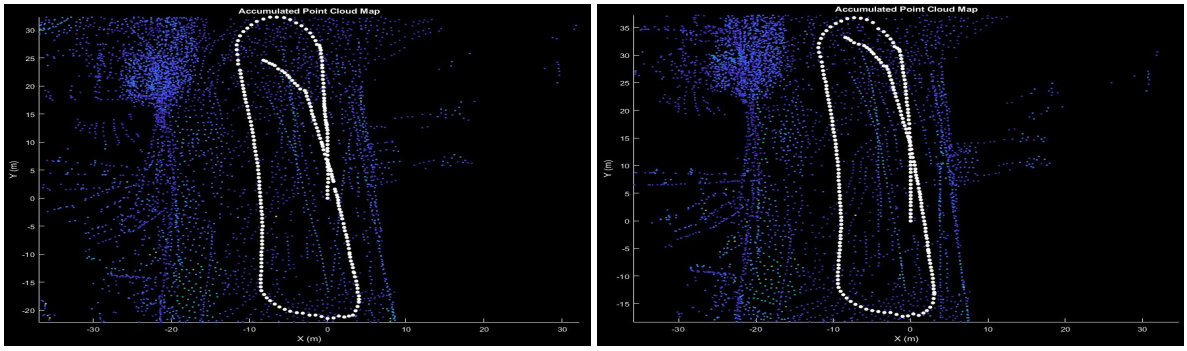
(a) *maxD=0.15.*



(b) *maxD=0.2.*

Figure 3.23: *NDT tests in oval circuit at low speed for different values of ground removal parameter.*

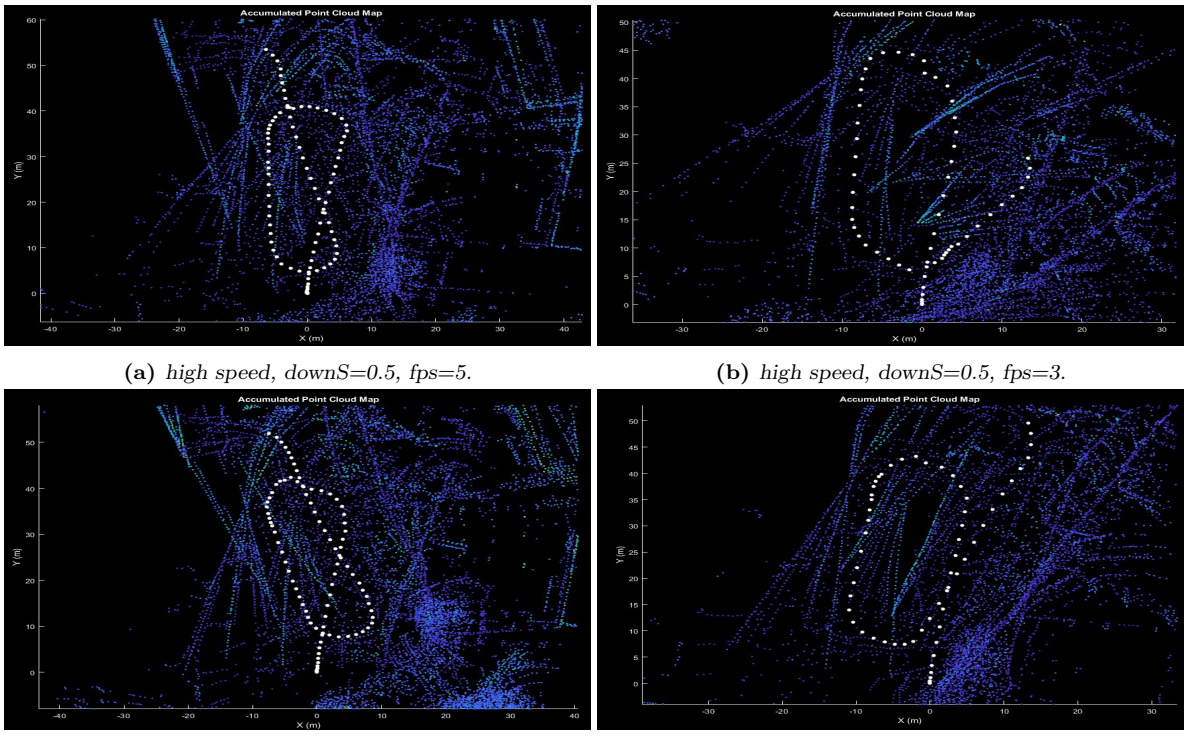
made using two 'ROI' defined by $xL = yL = 90$ and $xL = yL = 60$, the down-sampling equal to 0.95 and using 5 fps(Fig.3.24).



(a) *xL = yL = 90.* **(b)** *xL = yL = 60.*

Figure 3.24: *NDT tests in oval circuit at low speed for two reduced region-of-interest.*

In Fig.3.25 the high speed results are shown.



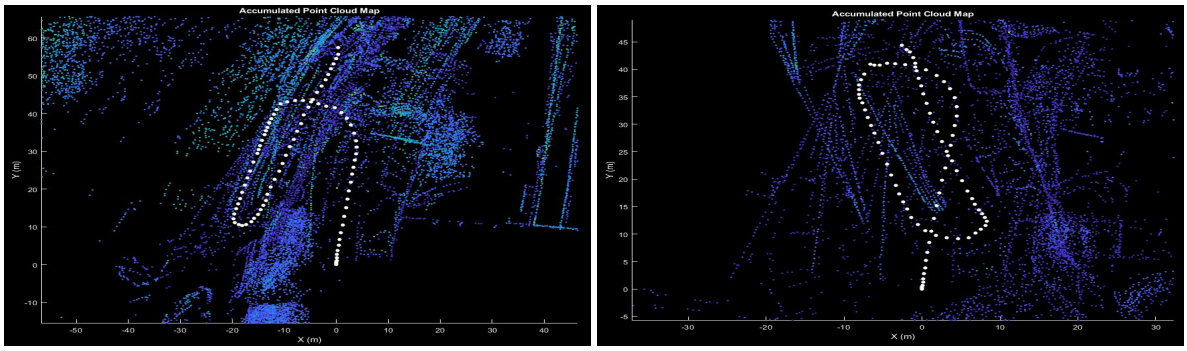
(a) *high speed, downS=0.5, fps=5.* **(b)** *high speed, downS=0.5, fps=3.*

(c) *high speed, downS=0.95, fps=5.* **(d)** *high speed, downS=0.95, fps=3.*

Figure 3.25: *Differences in changing the downsampling and the fps for a oval at high speed, in the outdoor data-set .*

Then the ground removal parameter 'maxD' has been changed with two different values as in Fig.3.26.

The last 3 tests have been done by increasing to 10 fps and by using the 'ROI' as done for the low speed case. Fig.3.27 and Fig.3.28 shows the results.



(a) *maxD=0.15.* **(b)** *maxD=0.2.*

Figure 3.26: *NDT tests in oval circuit at high speed for different values of ground removal parameter.*

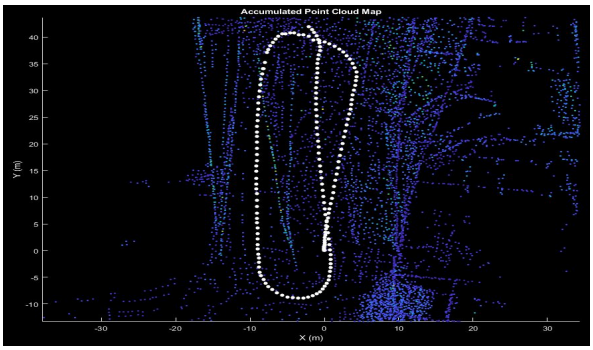
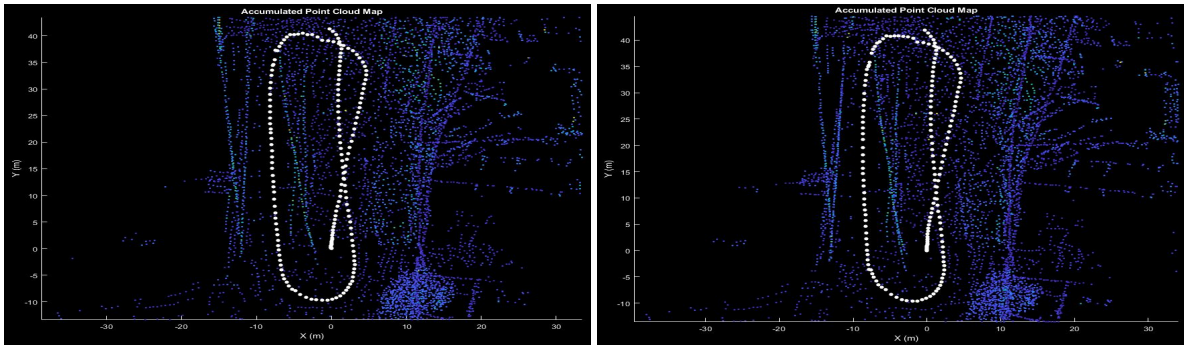


Figure 3.27: *NDT test for in oval circuit at high speed using 10 fps.*



(a) *xL=yL=90.* **(b)** *xL=yL=60.*

Figure 3.28: *NDT tests in oval circuit at high speed for for two reduced region-of-interest.*

Chapter 4

Discussions

This chapter contains comments about all the illustrated results of the previous chapter. In particular the discussion will address

- how the single change of a parameter can affect the results
- the reason why in some cases a different parameter has been change at the expense to the other
- if what we expect from the variation of a parameter is what we have obtained
- the main differences and reasons for changing a parameter in a data-set with respect to another data-set

All the discussed tests came from a 'trial and error' approach with some reasoning on what change and what could remain constant.

4.1 Testing scenario

The 3 different data-sets have specific aim in term of testing results. Below a brief description of each one of them:

- **Monterey-Highway**: this test scenario is not experimental, so this data have not been collected with our sensor. The commented results give us an idea of what can happen by testing similar data-sets using our sensor, the Velodyne VLP-16, keeping in mind that nothing is hundred percent known for example the speed, the traveled distance or the traffic in that road that can represent a sort of noise. So it can be seen as an urban road scenario from which we have obtained a first approximation of the parameter-set that can be used for future experimental acquisition with our sensor.

- **Indoor**: for this experimental scenario the two methods have been tested at a really low speed with any source of noise. This was a preliminary data acquisition that allowed us to make first assumptions and changes in the algorithm. The limits of this scenario are related to the small space at our disposal.

- **Outdoor**: this last experimental scenario is the most complete one because the data have been collected at different speeds and distances, with more turns and sources of noise. From this results the stability and robustness of the methods can be pointed out.

Taking all together the results and the peculiar things coming from that, these 3 scenario represent a good start for the development of this methods.

4.2 Results and discussion

In this section the results from the simulations are presented and discussed.
There is not a precision parameter on which we can rely, but the overall main trajectories are well known.

4.2.1 Monterey Highway

Starting from the results of the ICP method, the first thing that was pointed out is that changing both the down-sampling and the number of frames per second can be seen in different directions:

- if we increase computational speed with either less fps or more down-sampling, we get lot of information lost
- if we reduce computational speed with either more fps or less down-sampling, we get less information lost
- with less down-sampling we could not elaborate the points precisely, because of the method

Fig.3.3 and Fig.3.4 shows how the ICP method is bad in this situation, because the car should follow the trajectory rather, at a certain point, it is taking the left direction instead of continuing in the other lane and any further change in parameters make the result closer to the assumed trajectory. Indeed any change in the parameters doesn't led to a good trajectory.
Fig.3.12 and Fig.3.13 shows that the NDT algorithm is better in this case, because the expected trajectory is more or less obtained. So from our assumptions the trajectory (b) in Fig.3.12 is better than the (b) in Fig.3.13 because with more down-sampling we can have faster computation.
Usage of the region-of-interest was intended to reduce computation time and it may be seen as a little loss of data: starting from the assumption that between the left lane(negative y direction) and the right lane(positive y direction) there are approximately 15 meters, without using the region-of-interest the distance between the trajectory points in the x coordinate is in the range 15-20 meters, on the other hand at almost the end of the simulation(Fig.3.13) the car seems to go against traffic that is clearly a computation error.
In this case a typical source of noise that make the method fail in the transformation computation are all the moving entities like other cars, motorcycle and bike, so if the moving obstacles can be erased from the frame before to process it the mapping may

results better. Another important aspect is the unknown speed of the vehicle: the used algorithm is intended to work for a constant speed therefore in acceleration variations the method could be wrong where more complex environment is detected.

4.2.2 Indoor

In these tests, the ground removal parameters have been changed due to the different location of the sensor that is very close to the ground at more or less 10 cm from it. The maximum distance is set to 0.08 meter, the maximum angular distance is set to 10° and the radius is set to 1.2 meters. All the tests prove that using ICP or NDT does not make huge changes, indeed for both the small pieces of distance traveled no errors came out. The unique error that we can see comes from the still condition at the start and the stop of each simulation where a little deviation appears.
Due to the position of the sensor we expect to have noisy data when the speed increase because the front wing of the car is exposed to a lot of vibrations. This is not the case because the car has been pushed so very low speed doesn't affect the measurements.
From Fig.3.6 we can see that the trajectory is well defined also using less frames per second, but obviously the computational speed is better in the last case where the fps was equal to 1. For the (b) case where the number of frames per second was 2 the delimiting cones of the turn are all well defined in the mapping which may be useful in real-time application for example for the definition of left and right boundaries through the cones positions.
Fig.3.7 show how the mapping change with respect to the down-sampling, keeping constant the fps to 1: between the 3 cases there are no important differences in cones recognition, but we have to keep in mind that the speed is very low so for future high speed tests a low down-sampling may help to have a good result. Finally, for these data-sets, there is no method better than the other, but if we want to define it i will say that ICP is better: less deviation when the car is still and the walls are well detailed.

4.2.3 Outdoor

Also for these tests the sensor location was changed so also the ground removal parameter were changed as for the indoor data-set. So the maximum distance equal to 0.08 meters, the maximum angular distance equal to 10° and the radius equal to 1.2 meters. The 'Straight acceleration' regards a traveled distance more or less equal to 80 meters while the 'Ovale circuit' has two straight lines of almost 50 meters and the inner space cover more or less 5-7 meters on the x coordinate.
Now seeing the results coming from ICP(Fig.3.9, Fig.3.10, Fig.3.11) tests for both 'Straight acceleration' and 'Ovale circuit' there nothing more than bad results:

- changing the number of frames per second doesn't change anything and the mapped trajectory is neither a straight line nor an oval in any of the two cases, low and high speed
- any variation in the down-sampling has not been made because using the ninety-nine percent of the frame resulted in a failure, so it would have made any sense to reduce it

By the way in the 'Ovale circuit' data-sets variations in both fps and down-sampling have been made to actually see that this method in this cases doesn't return any good results.
For what concern the NDT method better results have been obtained. The 'Straight acceleration' tests at low speed in Fig.3.17 shows that reducing the number of fps the mapping returns what we expect to have in all the cases, so we have decided to keep constant the fps to 2 and then change the down-sampling to speed-up the process. As we can see in Fig.3.18 using the thirty percent the mapping doesn't change, but the computation is faster. In order to further increase the computational speed another test(Fig.3.19) has been made setting the down-sampling to 0.5 and the fps to 1 resulting in a bad mapping. In the best situation the mapping result in a straight line of more or less 80 meters so for this data-sets a good and reliable parameter set has been found. At high speed the results are quite different:

- Fig.3.20 show that decreasing the fps value to 1 results in a bad mapping while for 2 fps the mapping can be acceptable

- Fig.3.21 show that decreasing the down-sampling to 0.5 results in a bad mapping while for a lower down-sampling the map is good.

With respect to the low speed case now both the down-sampling and the fps can't be reduced in the same way to obtain a good mapping: the principal reason is the higher speed that doesn't allow to process frames that have lost too many data as required for the transformation computation.
The last tests have been made in the 'Ovale circuit' data-sets with two different results for low and high speed. First we wanted to see what happen having two degree of freedom, so changing both down-sampling and number of fps. From Fig.3.22 the first thing that can be seen is that reducing the fps the mapped trajectory is bad, instead for both fifty and five percent of down-sampling, at 5 fps, the trajectory is near the real one. All the tests have been made cutting the simulation after the first lap of the circuit for graphic reasons: we know that the algorithm is not precise so we wanted to point out how a single lap is mapped. Then two different approach to improve the mapping have been followed:

1. Fig.3.23: First by changing the ground removal parameter in order to reduce as much as possible the noise introduced by the bumpy road of the track, the case (b) is the best of the two

2. Fig.3.24: Second reducing the region-of-interest to be process by the algorithm because the very far points can be eliminated as a pre-processing phase in order to reduce, also in this case, the noise. Both of the two tests seems to be good in term of mapped trajectory

To choose if the mapped trajectory can be accepted or not we see if the two straight pieces of track are more or less 45-50 meters and if the distance between the point before the first left turn and the point after the first left turn is more or less 9-10 meters, the same is done also for the second left turn. From the last tests we can say that the trajectory is mapped with good precision, but without the help of a correction function, we expect to have a deviation after the two turns. Speaking about the cones, in all the simulations done we can't recognize them because of the noise first. Only when the simulation is starting the map returns well the position of the front cones but

then with all the points overlapped and with the error introduced by the transformation computation, it is impossible to recognize all the cones.

Finally in the high speed case of 'Ovale circuit' data-sets the same first tests done for low speed have been done: Fig.3.25 and Fig.3.23 shows how the change in down-sampling and number of fps and the change in ground removal parameter doesn't return any good mapping. The main reason, as it happens in 'Straight acceleration', is the increased speed of the car: any of the tests made by changing down-sampling or the removal parameter have returned an acceptable mapping. Then the idea was to increase the value of fps as shown in Fig.3.27 where the mapped trajectory is not clearly an oval but seems to be a first good approximation of that. The last two tests have been done following the idea used for the low speed case, so check the mapping through two reduced region-of-interest(Fig.3.28) maintaining the fps to 10: both the results are similar to the one without 'ROI'. In these final tests the mapping result to be acceptable because the straight lines, in the oval, are more or less of 45 meters while in the first left turn the distance in the x coordinate from before and after the turn is almost 12 meter, instead in the second turn the distance is more or less 10 meters. The principal source of error is the increased speed as already said.

Chapter 5

Conclusions and future works

In this thesis work, an algorithm for a sort of SLAM has been tested with two methods and in different test cases as high, low and very low speed, left or right turn, complete lap of a circuit. The algorithm has been tested in MATLAB using the different data-sets as explained in Chapter 3: Monterey-highway, Indoor and Outdoor data-sets where different results have led to our conclusion. For the Monterey-highway data-set, that is the one not experimental, we already said that the assumptions made are very strong and based on the streaming of the file, but from the initial tests on this data-set we were capable of make faster reasoning in changing the parameter-set for the other data-sets. So what i want to underline is that in this case the processed LiDAR data were full of noises and from this fact, my conclusions are:

1. in urban scenario where there are a lot of dynamic obstacles, then not a static environment, a simple rigid transformation for the mapping is not enough to have a good precision: a preliminary or complementary obstacle detection can be done to work side by side with SLAM

2. for sure the changing speed, so different acceleration during the acquisition, have to be taken into account

3. a well and defined knowledge of the boundaries make the mapping easier to obtain and more fast with less errors

In the Indoor data-sets no one relevant situation was met except for the still condition were there was a deviation error, so in this cases knowing that the speed is zero the mapped trajectory doesn't need to be updated while the surrounding has to.
In the last Outdoor tests a lot of considerations have to be pointed out because there are sources of noise, turns and straight lines, recognition of cones so it may represent, as already said, the best set to verify where the algorithm is good or not. The main problem that was encountered is related to the changing in speed that limits the number of fps that can be used for mapping: one possible solution for that is to have a dynamic value of number of fps that change proportionally with the speed, so at higher velocities more fps are needed while less fps when the velocity decrease.
Another problem may be the error in transformation computation for tight turns where the car change its heading of too many degrees: if the angle variation computation is not accurate, the little deviation error affect systematically the trajectory points. So with the help of an Inertia Measurement Unit this kind of error may be compensated introducing a minimization function in the heading angle estimation: taking as a reference the IMU heading data, if after the rigid transformation application the heading deviation from the reference is over a threshold the compensation start trying to minimize the error. Another possible solution is to use PID or MPC control strategies to follow the desired heading knowing the actual transformation. Future works may be integrate the IMU measurement data of orientation in order to compensate this error and refine the computation of the transformation.
The increasing speed so it's the most limiting thing for this algorithm that in some cases may not respect the real-time requirements. In this case for the future a dynamic approach in terms of number of fps proportional with the speed can be implemented, so from the IMU measurement data change the fps to avoid loss of performance: in case of positive acceleration(increased speed), the number of fps has to increase, while for negative acceleration(reduced speed) the number of fps may be reduced.
The problem of bumpy road as a source of error it's related to the ground removal pre-processing phase that, in our case, may be solved changing to an higher position the sensor location: in this way we can take advantage of all the 16 channels of the Velodyne, so the points belonging to all the lower points in the z coordinate of the

ground can be erased without affecting the precision of the transformation computation.

To conclude, this thesis work was done at the Mechatronic laboratory LIM and what I've done until now can be developed for future work and research in the autonomous driving field.

List of Figures

Bibliography

[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[2] Brent Schwarz. Lidar: Mapping the world in 3d. *Nature Photonics*, 4(7):429, 2010.

[3] MP Ananda, H Bernstein, KE Cunningham, WA Feess, and EG Stroud. Global positioning system (gps) autonomous navigation. In *IEEE Symposium on Position Location and Navigation. A Decade of Excellence in the Navigation Sciences*, pages 497–508. IEEE, 1990.

[4] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M Khairi, and Vijayabaskar Kasi. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262, 2013.

[5] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[6] Scott Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Eng, Daniela Rus, and Marcelo Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[7] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In *Robotics Research*, pages 613–625. Springer, 1996.

[8] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[9] Frank Moosmann and Christoph Stiller. Velodyne slam. In *2011 ieee intelligent vehicles symposium (iv)*, pages 393–398. IEEE, 2011.

[10] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[11] Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, et al. Team annieway's autonomous system for the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[12] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[13] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of field Robotics*, 25(8):467–492, 2008.

[14] Tao Mei, Huawei Liang, Bin Kong, Jing Yang, Hui Zhu, Bichun Li, Jiajia Chen, Pan Zhao, Tiejuan Xu, Xiang Tao, et al. Development of 'intelligent pioneer'unmanned vehicle. In *2012 IEEE Intelligent Vehicles Symposium*, pages 938–943. IEEE, 2012.

[15] Qingquan Li, Long Chen, Ming Li, Shih-Lung Shaw, and Andreas Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*, 63(2):540–555, 2013.

[16] Keqiang Li, Xiao Wang, Youchun Xu, and Jianqiang Wang. Density enhancement-based long-range pedestrian detection using 3-d range data. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1368–1380, 2015.

[17] MathWorks. Affine transformation. https://it.mathworks.com/discovery/affine-transformation.html.

[18] Francesco Amigoni, Simone Gasparini, and Maria Gini. Scan matching without odometry information. 2005.

[19] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[20] Cihan Ulas and Hakan Temeltas. A 3d scan matching method based on multi-layered normal distribution transform. *IFAC Proceedings Volumes*, 44(1):11602–11607, 2011.

[21] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam, 1996.

[22] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.