POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND
COMPUTER ENGINEERING

Final Project

3D Mapping of indoor environments using RGB-D Kinect camera for robotic mobile application

Supervisor:
Prof. Basilio Bona

Emmanuel Perez Bonnal

July 2011

Acknowledgements

I would like to specially thank to 3 people that have been a big help during the realization of this project. Thanks to Miguel Kaouk for all his counselling on theoretical and practical work, to Vito Macchia for his patience and always being available for any technical doubt or trouble and to Stefano Rosa for helping me to understand better the details of MMCL.

Contents

Chapter 1

Introduction

1.1 Point Cloud: A 3D Representation of the world

A point p_i, described by $\{x_i,y_i,z_i\}$, is a primitive representation unit in a three-dimensional Euclidean space. A collection of 3D points allows having a discrete representation of the surrounding world. We will refer to such 3D points collection as a point cloud P.

3D perception systems use point clouds as basic input data in order to store the information (e.g. in .pcd files). Every point p_i is given with respect to a fixed coordinate system. In most of the cases, this coordinate system has its origin at the sensing device used to acquire the data. For this reason, we can understand a point p_i as the distance on the three defined coordinate axes between the acquisition viewpoint and the surface from where the point has been sampled on.

Storing the 3D-position only could be not enough information for many applications. Geometric processing steps are necessary in many cases in order to extract meaningful information. For example, it is mandatory for partial alignment of two different point clouds to process and convert the raw input data several times into the representations and formats required by every individual processing step. More details will be given about this process in chapter 4.

Other processing steps such as segmentation and clustering can provide new properties as output to different points of a point cloud. For example, it is possible to assign different colors to identified objects in order to classify to which object pertains every point. Figure 1.1 represents a point cloud taken from a table, where each point has been colored according to its received class label. In this example, tables are colored green, object clusters supported by them are colored with a random color and remaining points are marked with black [Rus09].

Furthermore, RGB information can be used for a better understanding of the point cloud or for keypoint extraction to improve the registration process [HKH$^+$10].

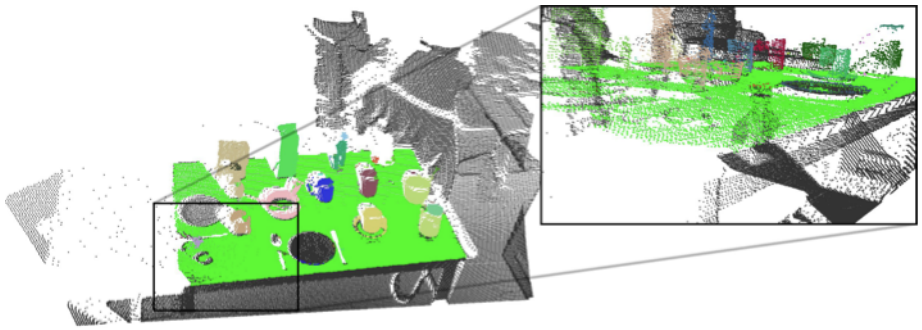


Figure 1.1. Regions of interest in a point cloud dataset, represented using different class labels and colors. (Source [Rus09])

A new definition of a point p_i is needed. Instead of only represent a position $\{x_i,y_i,z_i\}$ we will understand p_i as a set of features representing position, color, geometry, etc. A point p_i is then defined by $p_i = \{f_1,f_2 \cdots f_n\}$, meaning that it is now an nD descriptor instead of 3D [Rus09].

1.2 Objectives

RGB-D cameras are new, low cost, sensors that provide depth information for every RGB pixel acquired. Combining this information, it is possible to develop 3D perception in an indoor environment. In this paper we investigate how this technology can be used for building 3D maps. Such maps can gain more importance in the context of mobile robotics, as it can be used for many applications such as robot navigation.

In our work, we present 3D Localised Mapping (3DLM) an implemented algorithm that, knowing the camera's pose when acquiring the point clouds, is able to build such maps. We will investigate on two different ways, a simpler one and a more elaborate one, to process the reconstruction. In chapter 2 we will describe both hardware and software tools used for this project. In chapter 3 we will investigate on the point cloud acquisition process and in chapter 4 we will discuss about the 3D reconstruction process. Finally, in chapter 5 we will test 3DLM for a particular environment and analyse the results obtained.

Chapter 2

Tools

In this chapter we describe the different tools used in this project to implement the algorithm and run the tests. First, we will show the different hardware tools used and then we will discuss about different software libraries on which we based our work.

2.1 Hardware tools

2.1.1 RGB-D Kinect camera

RGB-D cameras are 3D sensing systems that acquire both RGB images and depth information for every pixel. It is possible to build 3D point clouds using depth information, which are well suited for 3D reconstruction and frame-to-frame alignment but ignore valuable visual information. On the other hand, RGB data can be more appropriate for other processes such as loop closure detection [NSS+09, KA08, MDH+09]. Combining both characteristics seems to represent an opportunity to develop more on robotics navigation and object recognition. There are many ways to obtain depth information in order to build 3D point clouds [N09], and most of the RGB-D cameras rely on Time-Of-Flight (TOF) sensing [Mes] or active stereo [Kon10]. TOF systems estimates the true distance from the camera to a surface by measuring the delay of an emitted signal returned to the receiver after hitting the surface.

The sensor used in our work is a RGB-D Kinect camera as in Figure 2.1, developed by PrimeSense, which acquires 640x480 registered images and depth data at 30 frames per second. The technology used for depth information is Light Coding which codes the scene volume with near-IR light. Then, a CMOS image sensor is used to read the coded light back from the scene and a depth image of the scene is produced after executing an algorithm to decipher the received light coding [Pri].

Kinect provides depth up to a limited distance of about 3.5 meters and its field of view is about 60° (see Appendix A.1). Though the official distance limit is 3.5 meters, Kinect actually acquires depth images of points being farther than this distance but there is a decrease of the cloud's accuracy.

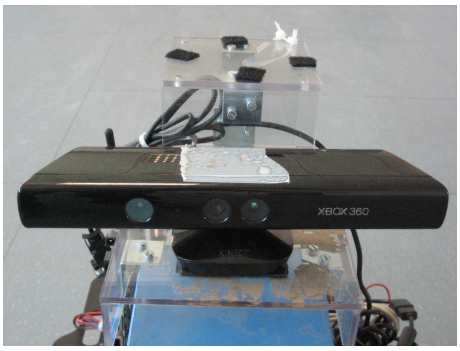


Figure 2.1. Kinect camera

2.1.2 Wheeled robot P3-DX

Figure 2.2. Rover P3DX

The robot used in our work is a MobileRobotics Pioneer P3DX as in Figure 2.2 (see Appendix A.2). It has an on-board PC that controls its motion using data acquired from a laser-range finder SICK LMS-200.

2.1.3 Laser Rangefinder SICK LMS-200

SICK Laser Measurement Sensor (LMS) 200 is the distance measurement sensor used (see Figure 2.3). It can provide distance measurements over a 180 degrees area and up to 32 meters away. In our case, the infrared laser is set up in order to realize 180 readings, so one measurement every degree, and has an operating range of 5 meters. The laser does not consider obstacles farther away than 5 meters. This laser is best for indoor environments as it can be dazzled by sunlight, causing erroneous readings. It is used in order to perform localization as well as obstacle detection (see Appendix A.3).

Figure 2.3. SICK Laser Measurement Sensor 200

Combining the three devices we obtain the operational rover of Figure 2.4 to run 3DLM.

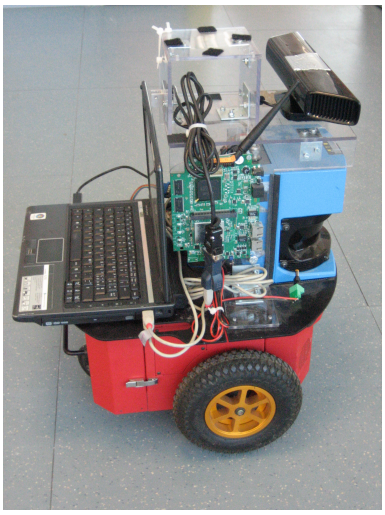


Figure 2.4. Operational rover used to run 3DLM

2.2 Software tools

2.2.1 Point Cloud Library

The main library used in our work for point cloud's processing is the Point Cloud Library [RC11]. It contains many state-of-the-art algorithms implemented, such as filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. It allows saving depth images captured by Kinect into .pcd files and processing point clouds.

2.2.2 Aria Library

MobileRobots' Advanced Robot Interface for Applications (ARIA) is a `C++` library (software development toolkit or SDK) for MobileRobots/ActivMedia platforms, in our case, a P3DX (see section 2.1.2). It provides an interface and framework for controlling robot's motion parameters such as velocity, heading or relative heading. ARIA can also receive odometric position estimates and other operating data sent by the MobileRobots platform. It provides also tools to integrate I/O with the robot and its accessories, in our case a SICK LMS-200 laser-range finder (see section 2.1.3). Furthermore, ARIA supports thread and network socket implementations, used in our work as explained in chapter 3.3.

ARIA is the base library used by MMCL explained below in order to manage P3-DX's motion and localization.

2.2.3 Multirobot MonteCarlo Localization

Multirobot MonteCarlo Localization (MMCL) is a program implemented by the Robotic's Search Group at Politecnico di Torino that allows to control several robots' motion and localise them. Based on a particle filter cooperative Monte Carlo Localization, it allows to localize different robots and track their position over time as they are moving in a highly symmetrical area [AIR+09, ABI+08]. As in our work there is only one robot, we do not take full advantage of this. However, our work is based on MMCL as it manages both P3DX's navigation and localization.

Navigation

P3-DX's laser sensor allows the robot to move around an indoor environment avoiding any obstacle bringing risk of collision. As explained in 2.1.3, the laser sensor has a range of 180 °, taking one reading every degree and with an operating range of 5 meters. In order to manage robot's navigation, MMCL allows to set its navigation behaviour in three different status: Wander, Teleoperation and Path planning.

Wander gives freedom of movement to the robot, going always straight until a potential obstacle is detected and then turning around until the detected obstacle is no longer more in its way. Teleoperation enables a complete control of its trajectory. The user can decide at any time which direction to take. Finally, by Path planning the user decides only the final position where the robot should go. The robot calculates the complete trajectory to realize and once it founds the best one, starts running toward the goal. Such calculations allows to have a smoother movement than the previous navigation modes. For this reason, we chose to use Path planning as navigation mode.

Localization

In order to be localized, the robot needs a 2D map of the environment. This map has to be done before starting to acquire the point clouds. Since there is no previous information about the environment before realizing the map, this task can be understood as a SLAM (Simultaneous Localization and Mapping) problem. To obtain this map, we have employed CoopSLAM, also implemented by Robotic's Search Group at Politecnico di Torino. CoopSLAM employs odometry to estimate its position relative to its starting location. This is possible thanks to rotatory encoders present in both robots' wheels. The laser acquires the data necessary to build the map. To reduce the error, it is mandatory to realize loop closure detection, which identifies when the robot has reached a position that has already been explored. Figure 2.5 represents a 2D map of a corridor of Politecnico di Torino's laboratory, obtained using the techniques described.

Figure 2.5. 2D Map of a corridor of DAUIN's laboratories

Once a 2D map is obtained, it is possible for the robot to localize its current

position in the map by comparing the current laser information with the laser information that should be detected in the estimated pose. The estimated position is actualized every 1.5 seconds. Between every pose update, the robot estimates its pose by using same odometry techniques than described above. Combining these techniques, it is possible to bind the error, preventing so an indefinite augment of it, characteristic of SLAM problems.

Chapter 3

3D Localised Mapping Implementation: Data Acquisition

The implemented algorithm 3D Localised Mapping (3DLM) consists of two main and independent processes. This chapter describes the structure of its first, Data Acquisition. Figure 3.1 shows the actual main structure of 3DML. In a first step, the data is acquired by a RGB-D camera while the rover is moving. In a second step, the data is processed, obtaining a 3D reconstruction of the environment.

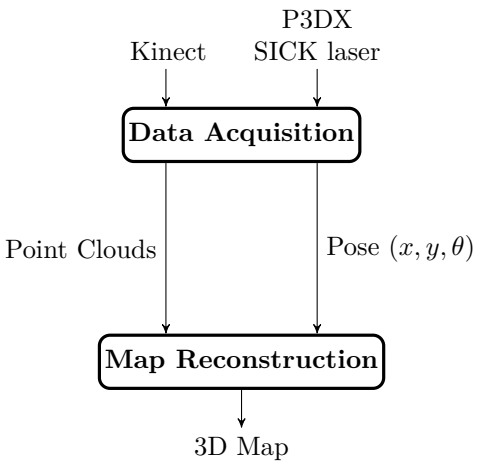


Figure 3.1. Global structure of 3D Localised Mapping

In order to reconstruct a 3D map, it is mandatory to obtain several depth pictures of the indoor environment and the pose from where they were acquired. Before, starting acquiring any data, the rover has to be localised in the 2D map obtained previously (see section 2.2.3). This means that an initial exploration has to be done until MMCL has correctly localised it. Once Data Acquisition has started, two

different processes run in a parallel matter. Figure 3.2 shows how Data Acquisition is organised. Cloud Acquisition saves frames from Kinect camera and MMCL manages both motion and localisation of the robot, registering the pose when it is required.

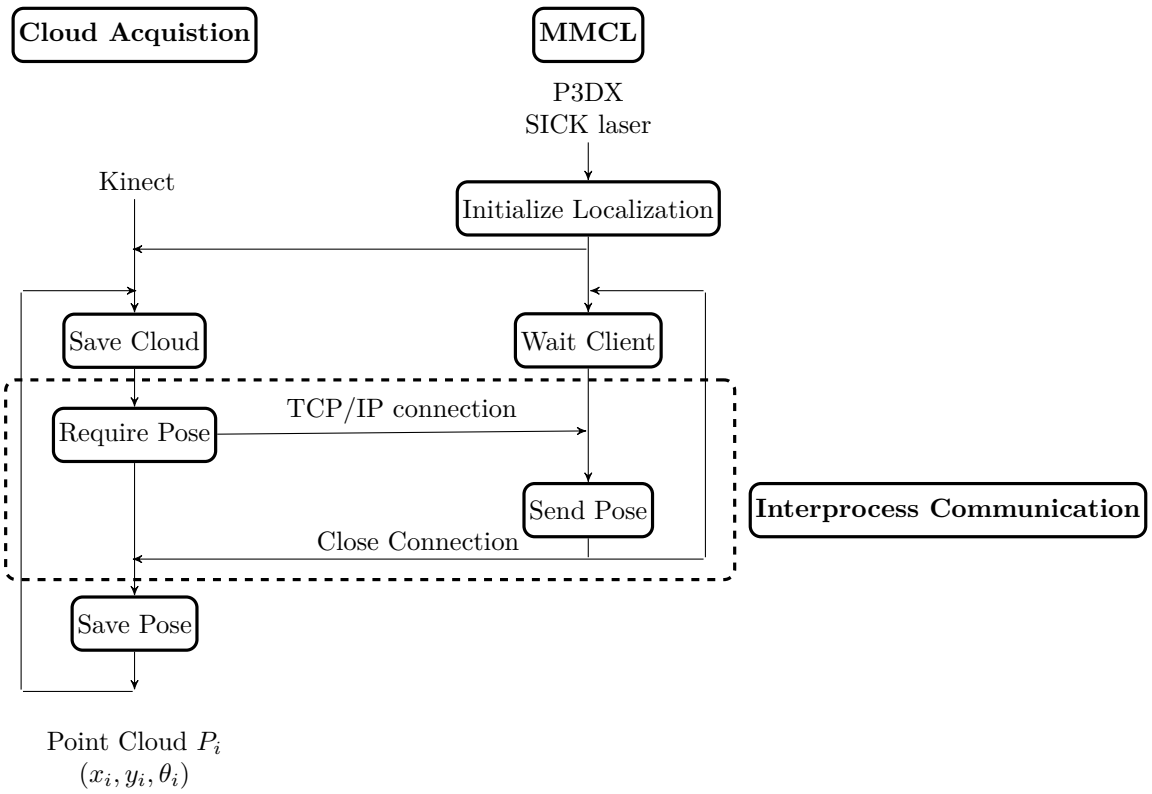


Figure 3.2. Data Acquisition structure

3.1 Cloud Acquisition

Kinect camera receives clouds at 30 frames per second. Therefore it is possible to regulate the acquisition's frame rate by setting Cloud Acquisition to save a cloud every n frames available .

Once a new cloud is saved, Cloud Acquisition requires to MMCL to send back the current robot's pose. It waits until the message is completely received and saves it in a text file, so that every new line corresponds to a new point cloud saved. In order to realize this task, it has been necessary to set a way in which Cloud Acquisition process and MMCL could communicate. Communication between both threads has been configured using socket as it is explained below. After saving the

pose, the process sleeps until next $n-1$ frames have been received and then starts saving the following cloud.

3.2 Acquisition of robot's pose

As explained in 2.2.3, MMCL estimates the position of the robot. When Cloud Acquisition requires the pose, MMCL saves last estimated pose and sends it if the robot is localized. If the rover is not yet localized, the pose is not sent.

3.3 Interprocess Communication

Sockets are I/O interface allowing communication between different processes even if they are running through different computers. They use TCP/IP connection, among others, to establish a link between them. Every machine running different threads receives an IP address and every process can be associated to a port in order to be identified. Assigning the correct IP, it is possible to identify a socket as coming from a local machine, which allows connecting different threads running in the same computer.

In our work, communication between both pose and point cloud acquisition's processes is of type server-client. We define MMCL as the server, waiting at every instant for the client to request robot's current pose. Point clouds' acquisition process is considered as client. Every time the client has acquired a new point cloud, it connects to the server, who immediately saves the current estimated pose and sends it back to the client. Once the message is completely received, the client closes its connection with the server. Both client and server are considered synchronous tasks as they block their processes until the communication is complete. This is not a problem as the communication process delay is usually under 100 ms and there is only one robot, so one client. However, if more than one client is required, a new implementation of the server should be done in order to convert it to asynchronous, giving it the ability to receive multiple clients at a time. Moreover, clients could be implemented as asynchronous if a faster acquisition of the point clouds was required, for example, to perform online the 3D reconstruction. This way, time wasted by the client waiting for the server to respond could be used. In our case, as we acquire 2 clouds per second and Map Reconstruction is done offline, it is not necessary to have this kind of client.

Chapter 4

3D Localised Mapping Implementation: Map Reconstruction

Once the entire data is saved, Map reconstruction starts to process this data in order to obtain the final map. This is done in two main processes. First, every cloud is filtered in order to reduce its size. Second, clouds are added to the map by registering them.

4.1 Cloud's filtering

Clouds acquired with Kinect camera have around 300 000 points. Clouds of this size require big computation times, use much memory but do not offer great advantages as, in our case, such precised clouds are not required. Filtering Cloud is done in order to reduce the use of resources and computational time. Figure 4.1 presents the different processes made to this purpose. It involves four steps: Pass Through filter, Down Sampling, Removing Outliers and Surface Reconstruction using Moving Least Squares. These methods are implemented in [RC11]

4.1.1 Pass Through filter

This first filter is used to set a depth limit in the cloud. Kinect camera has an operational range of 3.5 meters (see Appendix A.1). However, it actually acquires points that are farther than this distance threshold. We have noticed that in order to perform a correct reconstruction, it is necessary to use points being farther than 3.5 meters, even though depth's precision decrease as the distance between points and the camera grows. For this reason, Pass Through eliminates any point whose

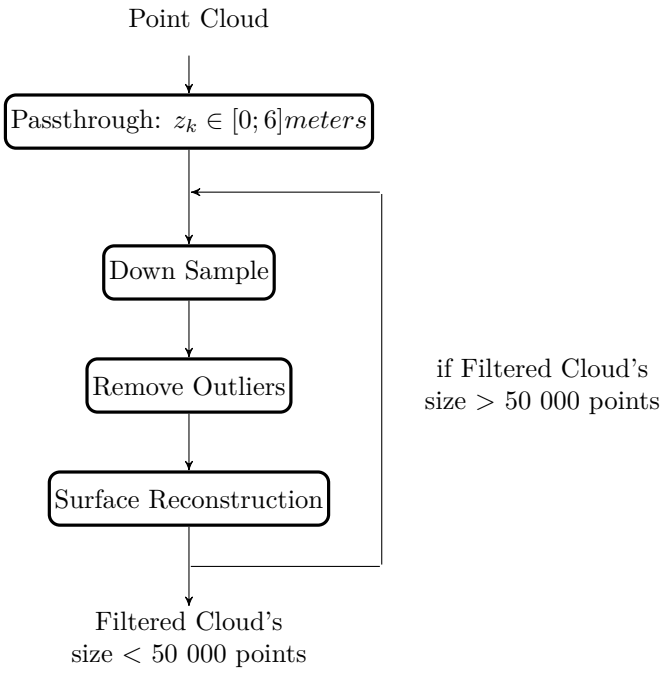


Figure 4.1. Steps to filter a point cloud

depth is farther than 6 meters from the camera. So, once Pass Through is done, all remaining points have $z_k \in$ [0;6] meters, z_k being the axis in Kinect's coordinate system representing depth.

4.1.2 Down Sampling

Two points pertaining to a point cloud taken by Kinect can have a minimal distance of about a few millimeters. In our case, mapping environments of several m^2, it is not mandatory to have such an accurate precision. It is so acceptable to lose some precision in order to reduce all point clouds' size, reducing considerably both computational time and memory usage. Down sampling reduces the number of points by using Voxel Grid. It divides a point cloud in boxes ("voxels") with the desired width. Then, all points within a box are reduced to an unique point corresponding to their centroid. This way, it is possible to set the minimum distance between points to the desired precision, reducing the number of points of a point cloud. In our work, voxels adopted are cubes of 2 cm of side.

4.1.3 Remove Outliers

Point clouds taken from Kinect can have measurement errors that create sparse outliers. Such points can lead to errors while estimating local point cloud's features such as surface normals. Calculations of this kind usually require to investigate a certain number of neighbours in the vicinity of a point, so it is necessary to ensure that the neighbours are correct. Moreover, removing such points will contribute to reduce processing time as it decreases the remaining point's number, although its effect in this matter is less important than Pass Through filter and Downsampling.

The method used to remove outliers is presented in [Rus09] and is based on performing statistical analysis on each point's neighborhood. Mean distance from each point to all its neighbors is calculated. Assuming a Gaussian distribution, with a mean and a standard deviation, all points having their mean distances outside a certain interval are considered as outliers and are removed from the dataset.

In our work, 50 neighbours are used for each point to analyse its status, removing all points having a distance larger than 1 standard deviation of the mean distance to the analysed point.

4.1.4 Surface Reconstruction

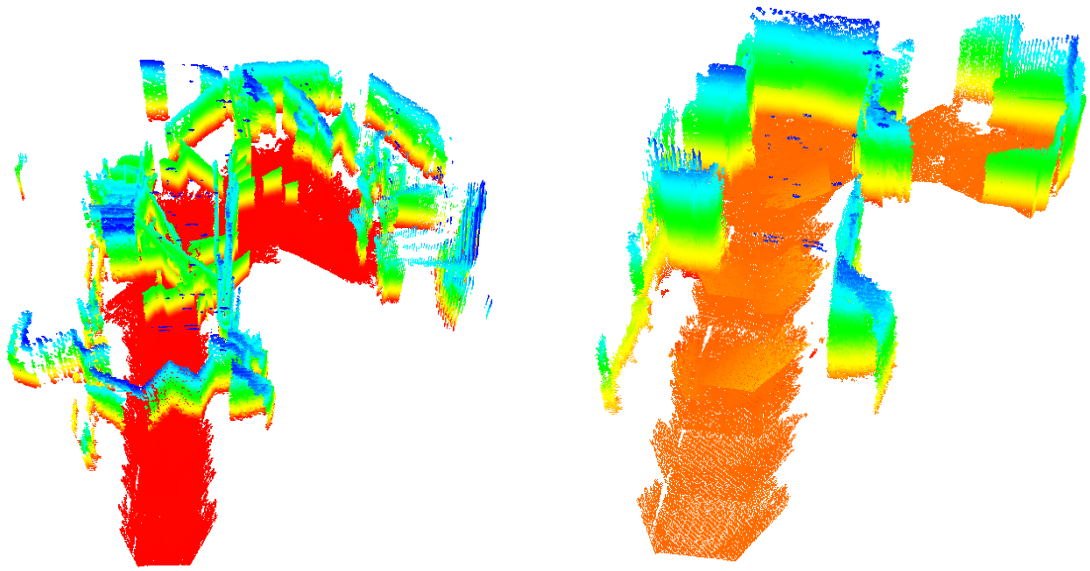
Surface reconstruction is used in order to improve removal of data irregularities. It is based on a Moving Least Squares (MLS) algorithm presented in [Rus09]. MLS provides a reconstructed surface for a given set of points by interpolating higher order polynomials between the surrounding local neighbors. Smoothing and resampling a noisy cloud allows to obtain more accurate estimations of surface normals and curvatures. Such estimations are used further to register point clouds (see section 4.2.4).

4.1.5 Filtering results

By performing these steps, the number of points within a point cloud is significantly reduced. In our work we have determined to use point clouds with less than 50 000 points. This is generally obtained after following the whole filtering process. However, in some cases point clouds still have a greater size than the accepted. In these cases, a second filtering process is realized. Reducing the number of points to such limits speeds up the time employed for alignment process, prevents from errors and presents a sufficient precision at the same time.

4.2 Registration process

Having robot's pose would allow, in theory, to reconstruct the whole map applying roto-translation to every cloud. However, the pose obtained by MMCL has an error that cannot be ignored. Figure 4.2 shows how applying roto-tranlation only and adding all clouds would leave to a map having plenty of errors.



(a) Direct sum

(b) Sum after registration process

Figure 4.2. Map resultant of adding clouds after a single roto-translation process (a) and after registration process using SAC-IA and ICP (b)

In order to improve the results, a registration process is required. We have studied two ways to realize this. A first process has been implemented using Iterative Closest Point (ICP) as unique alignment process. Further, we have built a second process applying two different registration processes, Sample Consensus Initial Alignment (SAC-IA) and ICP. In both cases, the methods have a structure corresponding to Figure 4.3. After initializing the map, for each new cloud we register it with the previous map and decide if the registration is good enough to include it to the map.

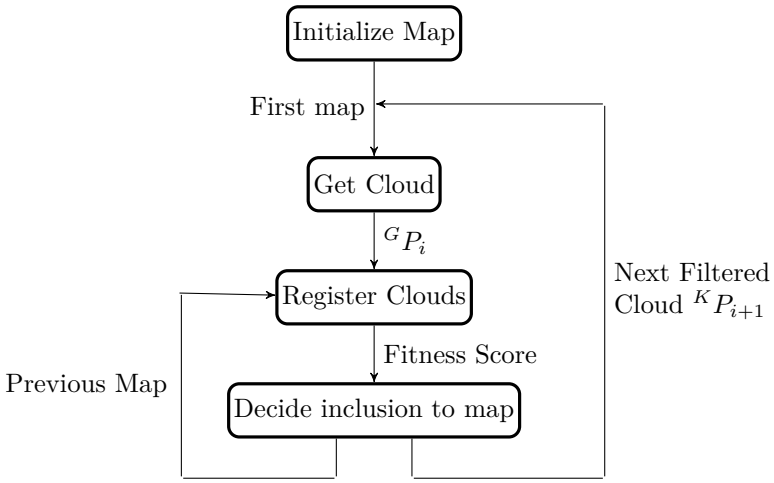


Figure 4.3. Registration process leading to a reconstructed map

4.2.1 Initialize Map

This process reads the first PCD file saved by Filtering Cloud (see section 4.1) and its pose from the first line of the text file saved by Cloud Acquisition during Data Acquisition (see section 3.1). It applies then a roto-translation as explained in 4.2.2, transforming the cloud from its local reference to the global reference. We will refer to ${}^{K}P_i$ as the i-th point cloud in its local reference and to ${}^{G}P_i$ as the same cloud in the global reference. At this point, the cloud is ready to be set as the first map with which any new acceptable cloud will be added. Figure 4.4 shows the steps followed by this method. In addition, if SAC-IA is used, calculation of cloud's features explained in 4.2.4 is done.

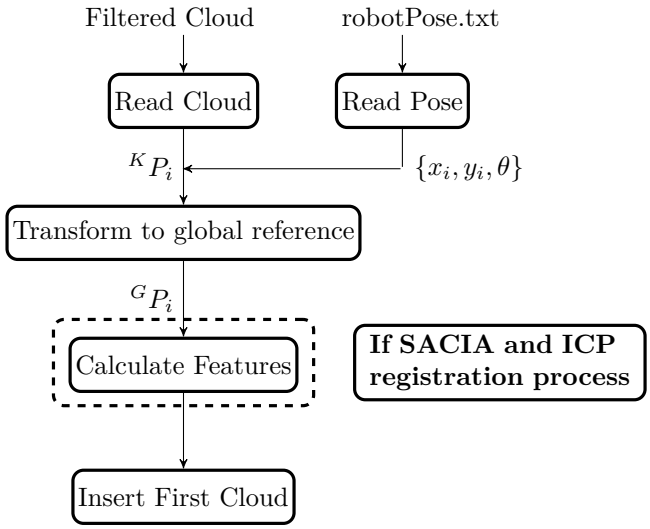


Figure 4.4. Structure of map's initialization

4.2.2 Get new cloud

For every new registration process, a new cloud is obtained reading next filtered cloud and the corresponding global pose of the camera from when the cloud was acquired. Using this information, it is possible to convert this cloud to the global reference.

From local to global reference

In order to combine different clouds and create a general map, they must all be in the same reference. Using the pose given by MMCL, it is possible to transform all points within a point cloud to a global reference. Figure 4.5 presents the two steps composing transformation process.

Acquired point clouds are referenced to the Kinect camera's coordinate system R_{kinect}. Its origin is located at the camera's sensor point and its axes x_k, y_k and z_k are defined as in Figure 4.6. It corresponds to a mobile coordinate system as the rover moves around during the acquisition of different clouds, changing the position of the camera. Robot's reference system R_{robot}, represented as well in Figure 4.6, is also a mobile system. We consider that origins of both R_{kinect} and R_{robot} are coincidents.

In order to adapt the cloud to the rover's reference system we apply the transformation T_K^R that allows to transform a cloud from Kinect's coordinates system to R_{robot}.

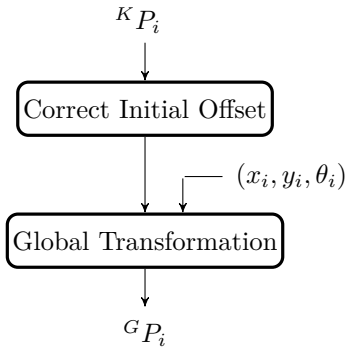


Figure 4.5. Transformation process of a point cloud referenced to Kinect ${}^{K}P_i$ to a Global reference obtaining ${}^{G}P_i$

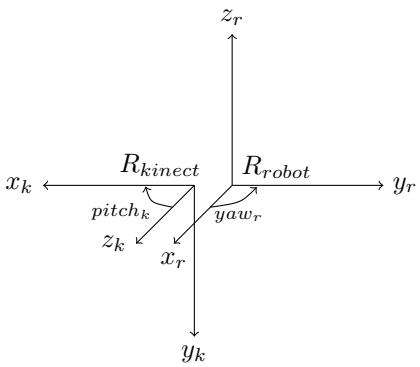


Figure 4.6. Coordinate systems of Kinect camera and rover

Once the cloud is in an appropriate local reference system, it is necessary to transform the cloud to a global reference system that will allow to work with other clouds. R_{global} is a fixed coordinate system that has the same origin and orientation than R_{robot} when rover's pose is (x_0,y_0,θ_0) = (0,0,0). P3DX has 3 degrees of freedom as its z_G coordinate and its $pitch_G$ and $roll_G$ do not change. This is due to the fact

$$T_K^R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.7. Matrix transformation T_K^R relates Kinect's coordinate system to the robot's coordinate system

that the rover moves around the floor, so the plane $x_G - y_G$, and that the camera does not move with respect to R_{robot}. Figure 4.8 shows the relation between the three different coordinate systems.

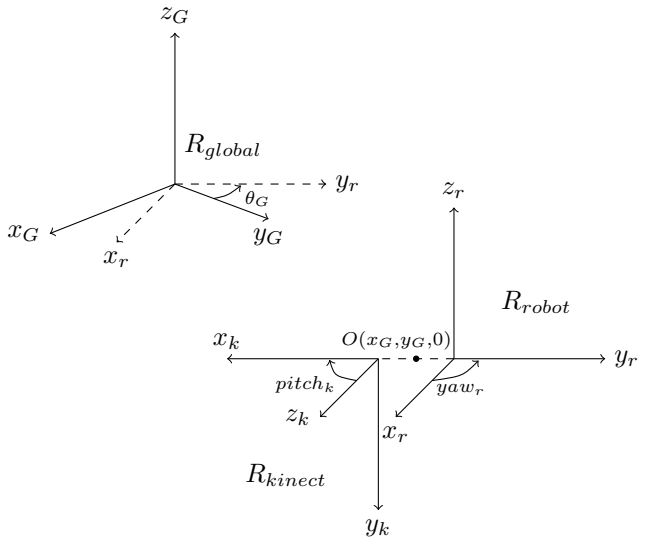


Figure 4.8. Global coordinate system and relative robot's and Kinect's systems

To transform the cloud referenced from its relative pose system R_{robot} to R_{global} we apply T_R^G transformation.

$$T_R^G = \begin{bmatrix} cos(\theta_G) & -sin(\theta_G) & 0 & x_G \\ sin(\theta_G) & cos(\theta_G) & 0 & y_G \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.9. Matrix transformation T_R^G relates robot's coordinate system to the Global coordinate system

The resultant transformation matrix that converts clouds acquired from Kinect to a global reference T_K^G is then the product of T_R^G and T_K^R.

Initial offset

We have affirmed that camera's and rover's coordinate systems are oriented in a way that x_r always corresponds to z_k. This might not always be correct. Kinect has actually one degree of liberty that allows to change the orientation of z_k. This

$$T_K^G = T_R^G \cdot T_K^R = \begin{bmatrix} sin(\theta_G) & 0 & cos(\theta_G) & x_G \\ -cos(\theta_G) & 0 & sin(\theta_G) & y_G \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.10. Matrix transformation T_K^G relates Kinect's coordinate system to the Global coordinate system

movement has to be done manually and consecutive orientation positions differ of about 6°. In case z_k is not parallel to the floor, and so to x_r, an initial correction of this offset has to be realized. This correction corresponds to a simple rotation of ω_k degrees around x_k and its transformation is defined as T_{offset}.

$$T_{offset} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\omega_k) & -sin(\omega_k) & 0 \\ 0 & sin(\omega_k) & cos(\omega_k) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.11. Matrix transformation T_{offset} corrects possible initial offset

4.2.3 Register Clouds: ICP alignment

The first registration process designed is described by Figure 4.12. It consists of a unique alignment between the current map and the new point cloud ${}^G P_i$ willed to be inserted in using the Iterative Closest Points (ICP) algorithm implemented in [RC11].

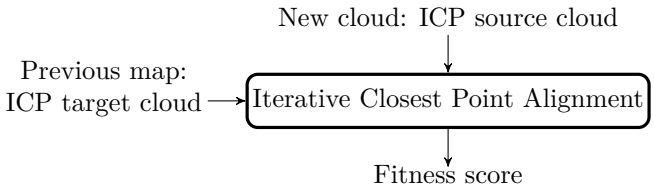


Figure 4.12. Registration process using only ICP

Iterative Closest Points

The Iterative Closest Points algorithm is a well known process that aligns two point cloud datasets by minimizing the Euclidean distance between their correspondent points [BM92, Zha92]. It finds pairs of nearest 3D points in the source and the target and defines them as correspondent if their distance is smaller than a specified distance. It then estimates a rigid transformation that minimizes the distance between them and iterates until the difference between consecutive transformations is smaller than the limit defined or until the maximum number of iterations is reached.

However, ICP can also have some problems as it converges to a local minima, so might give not a good registration result and it can need a great number of iterations.

The ICP function returns a parameter called fitness score that provides information about the quality of the alignment done. Fitness score corresponds to a calculation of the distance error between clouds aligned after the registration process. This parameter will be useful to decide in which cases the alignment is good enough to include it in the map.

4.2.4 Register Clouds: SAC-IA and ICP alignment

This second registration process attempts to obtain a more accurate alignment than the explained previously. After aligning the new cloud using ICP as in 4.2.3, this process realizes a second alignment process if the result is not acceptable. In this case, it uses SAmple Consensus Initial Alignment (SAC-IA) to obtain an initial alignment that will give a better first guess for ICP. After the second ICP is done, the results are given in order to decide either to include or not the new cloud to the map. Figure 4.13 presents the structure of the explained registration process.

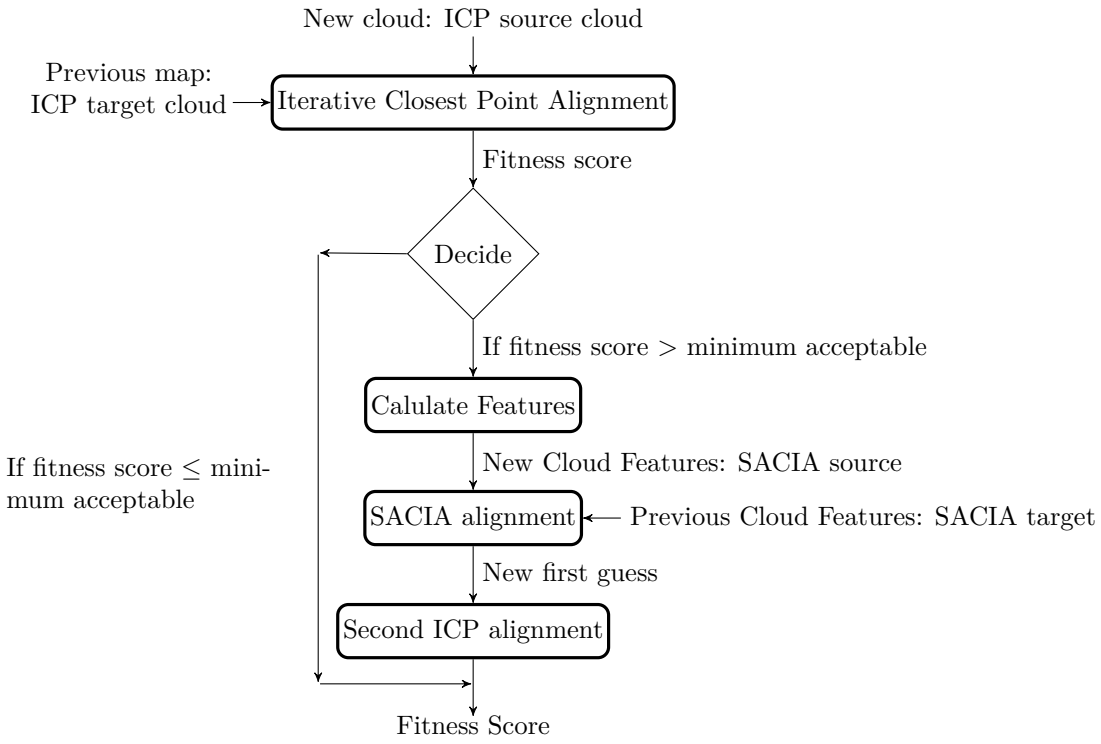


Figure 4.13. Registration process using SAC-IA and ICP

Calculate Features

In order to realize a SAC-IA alignment it is necessary to calculate first some geometrical features that will allow SAC-IA to proceed. Figure 4.14 shows the different steps followed to obtain the necessary features.

Reduction of Point Cloud's size

The time employed to realize features calculations and, more importantly, the SAC-IA alignment itself increases considerably as the size of a point cloud ${}^{G}P_i$ is greater. SAC-IA alignment between two point clouds of about 50 000 points each can take between 5 and 7 minutes, depending on the parameters used, while alignment of point clouds of about 3000 points takes 5 seconds. For this reason, the first step before calculating the desired features is to reduce its size. Moreover, instead of registering a new cloud with the current map, we apply alignment between the new cloud and the last cloud inserted in the map. This is done because as the map grows, its size increases too, slowing the alignment.

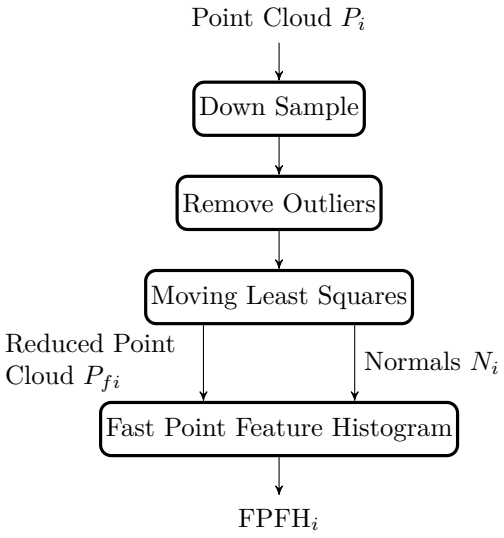


Figure 4.14. Process to obtain features needed for SAC-IA alignment

To reduce a point cloud's size, we apply the same operations than in 4.1: down sampling, removing outliers and resampling to smooth its surfaces. In this case, down sampling is set to use voxels of side 6 cm. The point clouds used for SAC-IA have all less than 4000 points. Additionally, surface reconstruction allows to obtain simultaneously more smoothed surfaces and the first feature needed, the normals estimation.

Surface Normals estimation

To estimate the normal direction would usually be trivial if a surface was given. However, a point cloud describes surfaces by a set of point samples of the real surface. To estimate the normal at every point $p_q \in P_i$, it is mandatory to analyse its neighbors p_q^k. Selecting p_q^k correctly is an important step in order to obtain a good estimation.

Two procedures are described in [Rus09] to determinate p_q^k. The first one is to determine the closest k neighbors to p_q (k-search) and the second one is to specify a certain radius r and include to p_q^k any point being closer to p_q than r (radius search). In our work, we have used this second method because k-search needs to estimate and order all distances from p_q to all the other points pertaining to P_i and so operates slower than radius-search. Moreover, using radius search will allow to estimate features independently on the number of points sampled. In our work, as we down sample the cloud up to 6 cm, we use a radius $r_{normals}$ of 20 cm to find p_q^k to estimate p_q's normal.

To estimate the normal of p_q, the function used and explained in [Rus09] attempts to determine the normal of a plane tangent to the surface formed by p_q^k.

Fast Point Features Histograms (FPFH)

The actual feature used by SAC-IA to perform alignment is FPFH. This feature uses the previously estimated normals to calculate some geometrical relations between a point p_q and its neighbors p_q^k. FPFH is described in [RBB09, Rus09] and attempts to create a multi-dimensional feature space that will allow to group in a same class all points sampled from the same surface while points pertaining to different surfaces would be assigned to different classes. FPFH estimation works as well with query point's neighbors p_q^k but demands to have a radius search greater than the one used for normal estimation. In our work, the second r_{FPFH} used is so 30 cm.

SAmple Consensus Initial Alignment

SAC-IA is an algorithm that uses FPFH to realize a first alignment between two different point clouds P and Q. It is presented in [RBB09] and is composed of three principal steps.

It selects s sample points from P making sure that their pairwise distances are greater than a certain minimum distance d_{min}. In our work, this distance is set to 40 cm. Then, it finds a list of points in Q whose FPFH histograms are similar to the sample points' histogram for each of the sample points. It selects one randomly from the list and this point will be considered as that sample points' correspondence. Finally, it computes the rigid transformation defined by the sample points and their correspondences and computes an error metric for the point cloud that allows to find the quality of the transformation.

In our work, we set the maximum number of iterations up to 1000 times and save the transformation giving the best error metric.

Final Alignment with ICP

After SAC-IA is done, the resultant transformation is used to transform the cloud wanted to be aligned with the map. Thereby, a second initial guess has been founded and ICP is computed again as in 4.2.3 in order to find an alignment that will give an acceptable fitness score.

4.2.5 Decision of the inclusion

In this final step, Registration process decides whether to include or not the transformed cloud to the map. To do so, we use a parameter given by ICP, registration's fitness score. We have found empirically that clouds having a fitness score minor

than 0.01 are sufficiently well aligned to the map and are so included. The Decision process is shown in Figure 4.15.

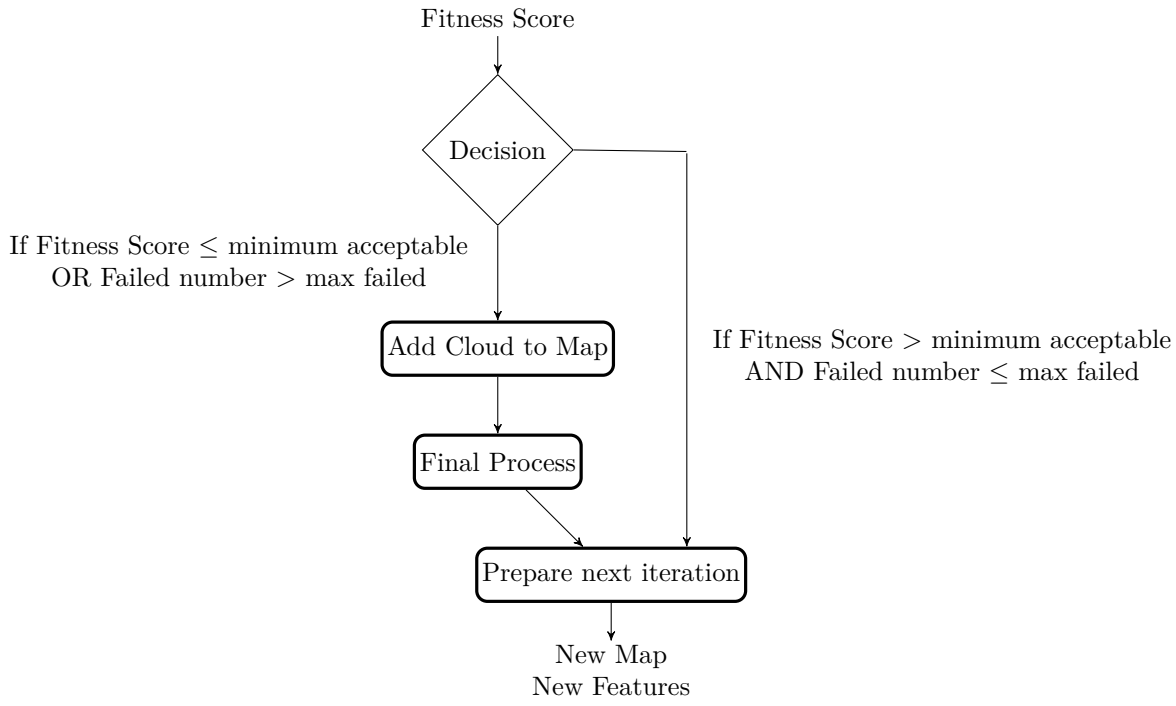


Figure 4.15. Decision process to include a registered cloud to the map

As registration process uses overlapping areas between point clouds to perform the alignment, such overlapping areas need to be always present. However this might not always be possible. Indeed if many consecutive clouds are not accepted, the next cloud might not have any common area with either the last cloud included or the map, resulting in that new explored areas will no longer be included in the 3D reconstruction. We decide to include a cloud with an unacceptable fitness cloud if the last 3 clouds have not been accepted. Instead of using the cloud transformed by ICP, we insert the initial ${}^{G}P_i$ in the map by simply adding its points to it. This is done because we have noticed that bad alignment results are usually worse than the initial guess. Therefore, we are assuming the risk of this cloud not being good in order to not be blocked and always be able to expand the reconstruction.

Final Process

If a new cloud is included, all its points are added to the previous map. In this case, overlapping areas will be more dense as it will contain two points for the same

estimated position. It is not necessary to have such density and it can also provide imprecision as the alignment between clouds is not without error. Therefore, a final filtering process is needed in order to maintain the coherent density. The map is down sampled using Voxel Grid again, setting it's size to 2 cm. Thereby the map is filtered to the same parameters as before the new cloud was added.

Prepare next iteration

As final step, 3DLM proceeds to set the next iteration. If a new cloud has been added, it is set as target for next ICP registration process and failed clouds counter is set to 0. If SAC-IA is used, local features of the last included cloud are calculated and set as target for next SAC-IA alignment.

If the cloud has been rejected, nothing has to be done but increase the counter of failed clouds that will allow to know if there have been too many consecutive rejected clouds.

Chapter 5

Experimental Test: 3D Reconstruction of a corridor

The explained algorithm lead to a reconstructed map of the environment, though the quality of it highly depends on the parameters used. We have designed a set of tests in order to investigate in which cases 3DLM brings better results for a particular environment.

5.1 Description of the test

The chosen environment for this section is a corridor of DAUIN's department at Politecnico di Torino. The experiment consists in reproducing the same trajectory with the robot several times, changing the rover's maximum speed (v_{Rovmax}), its maximum rotational speed (v_{Rotmax}) and the saving cloud frame rate during the acquisition process (f_{Acq}). Table 5.1 shows the different values used.

	v_{Rovmax} (mm/s)	v_{Rotmax} (deg/s)	f_{Acq} (fps)
T1	100	20	1
T2	200	20	1
T3	100	40	1
T4	200	40	1
T5	100	20	2
T6	200	20	2
T7	100	40	2
T8	200	40	2

Table 5.1. Combination of parameters used for 3D reconstruction of a corridor

The path taken by the robot is shown in Figure 5.1 and represents a 30 meters

run in total. Four positions A, B, C and D, are defined in order to maintain the same path and to analyse the influence of straight/rotational movement. A-B is a section where the rover goes straight, so θ_G given by MMCL is almost invariable. B-C and D-B combine forward movement and rotation while C-D is almost only rotational. As explained in 2.2.3, Path planning brings a smoother rotational movement than teleoperation while it allows to manage the direction taken by the rover. For this reason all the tests will be done using Path planning.

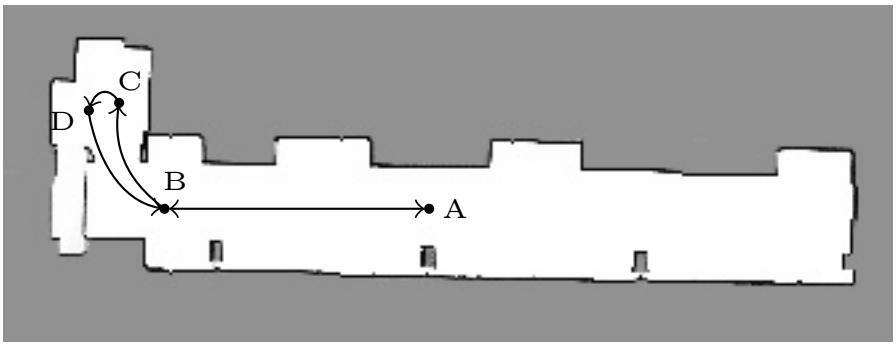


Figure 5.1. Path taken by the robot during tests T1-T8

5.2 Overlapping Area: Previous Calculations

We estimate that it is important, in order to obtain good alignment results, to have an acceptable initial guess. The quality of the initial guess relies on the data acquisition process. Clouds being registered need to have a sufficient overlapping area in order to obtain great results. Thus varying the speed of the rover/camera can modify the distance between clouds and their overlapping areas.

5.2.1 Straight Movement

When going straight, the distance between the origin of two consecutive acquired clouds δ is the distance between the positions from where the camera acquired them. Such δ depends on both rover's speed v_{Rov} and the acquisition frequency f_{Acq} as we can see in equation 5.1.

$$\delta = \frac{v_{Rov}}{f_{Acq}} \tag{5.1}$$

As in our test points farther than 6 meters from the camera are eliminated and the path is always in the x_G - y_G plane, we can consider that the overlapping area of two consecutive acquired clouds is characterized by $\sigma = 6 - \delta$ meters. Table

5.2 presents the values of σ for the chosen v_{Rovmax} and f_{Acq} and the percentage of overlapping area that this represents.

	v_{Rovmax} (mm/s)	f_{Acq} (fps)	σ (m)	Overlapping Area (%)
T1,T3	100	1	5.9	98.3
T2,T4	200	1	5.8	96.7
T5,T7	100	2	5.95	99.2
T6,T8	200	2	5.9	98.3

Table 5.2. Percentage of a cloud overlapped by next acquired cloud

In every case overlapping area represents a high percentage of the cloud so the influence of v_{Rovmax} seem not to be crucial.

5.2.2 Turning Movement

When turning, the camera remains at the same position, modifying only its orientation. The angle turned between two consecutive cloud acquisitions τ will influence in the overlapped areas between them. τ can be calculated using the rover's rotational speed v_{Rot} and f_{Acq} as in equation 5.2.

$$\tau = \frac{v_{Rot}}{f_{Acq}} \tag{5.2}$$

An observed surface being at a distance d from the camera will have a re-observed area after turning τ around z_G as in Figure 5.2.

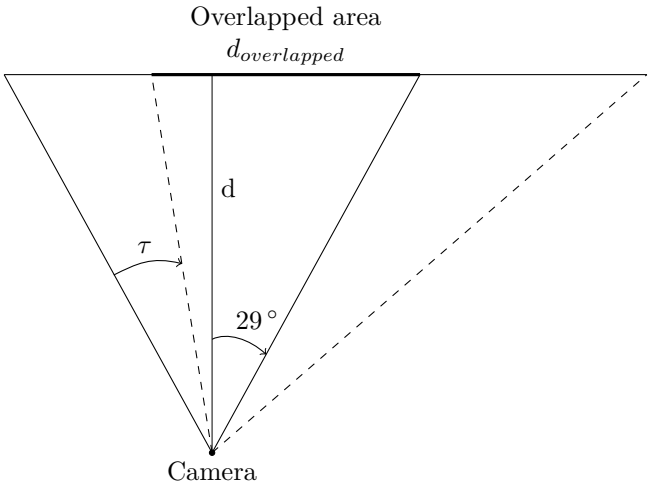


Figure 5.2. Overlapped area after a rotation of the camera of τ

Knowing that the Kinect has a field of view of 58 ° in the horizontal direction (see Appendix A.1) it is possible to find the area's width $d_{overlapped}$ of a surface, present in both clouds.

$$d_{overlapped} = d \cdot (tan(29) + tan(29 - \tau)) \tag{5.3}$$

Table 5.3 presents the estimation of the overlapped area's width between two consecutive acquired clouds when turning at v_{Rotmax}.

	v_{Rotmax} (deg/s)	f_{Acq} (fps)	τ (deg)	Overlapped Area (%)
T1,T2	20	1	20	64.3
T3,T4	40	1	40	32.5
T5,T6	20	2	10	81.1
T7,T8	40	2	20	64.3

Table 5.3. Percentage of overlapped area's width between two clouds

Observing the analysis made for forward and turning movement, we can estimate that rotation will have a stronger influence on the alignment results. We can also estimate that T1 and T8 will have a similar behaviour as their σ is 5.9 and their τ is 20 in both cases.

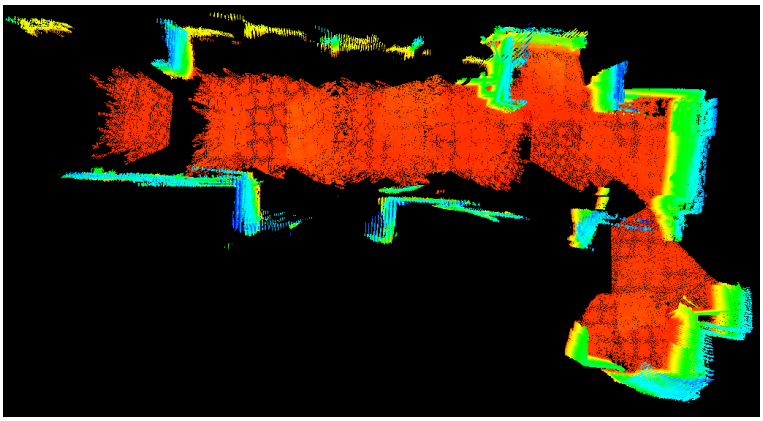
5.3 Parameters used

In order to investigate the influence of registering using ICP or a combination of SACIA and ICP, we process all the data using both options. Moreover, instead of doing twice the same path changing only f_{Acq}, we decide to take all the data at 2 frames per second, skipping an acquired cloud of two during the reconstruction process if the desired f_{Acq} is 1 fps. Thereby, the possibility of different results due to differences during data acquisition is avoided in these cases, allowing to obtain safer conclusions.

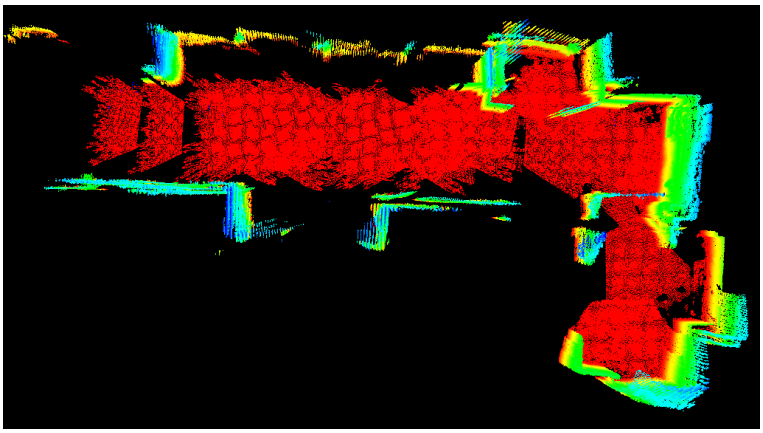
The other parameters used for every process remain constant for every process and can be found in appendix B

5.4 Results: Qualitative analysis

A first analysis can be done observing the maps obtained after every reconstruction. Figures 5.3 and 5.4 presents the resultant maps after running 3DLM using SACIA-ICP or only ICP respectively for two different tests. The shown results are from tests T2 and T5 which are respectively the worse and the best maps obtained.



(a) T2 using SACIA-ICP

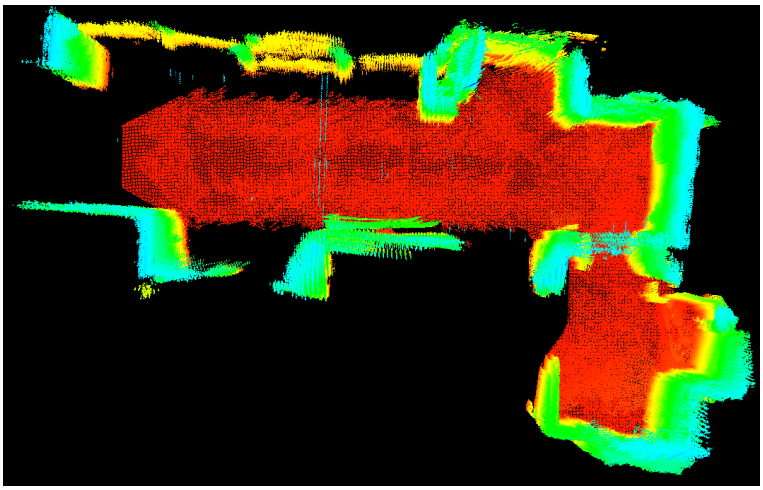


(b) T2 using ICP

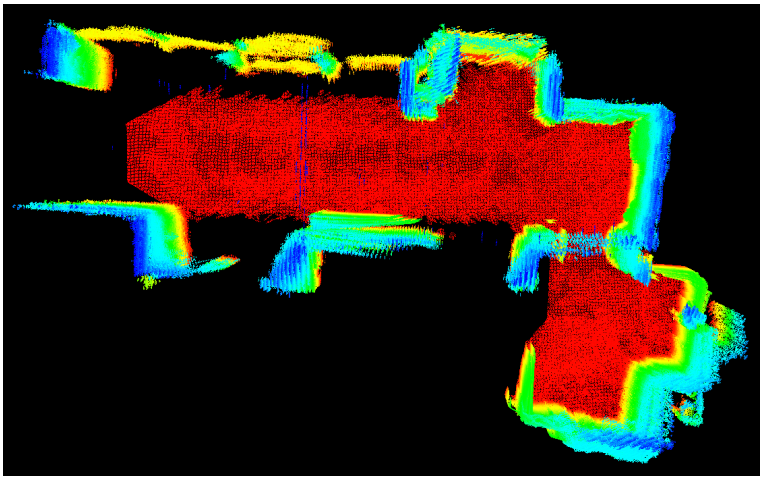
Figure 5.3. Reconstructed maps obtained for test T2 using SACIA-ICP and ICP

In Figures 5.5 and 5.6 we can observe some details of the different resultant maps. For each case, Figure 5.6 shows the detail of a column being between position A and B (so being seen from one side during A-B course and from the other side during B-A course). Not a single test has succeed to completely reconstruct the column, but we can see that the reconstruction of T5 is better than T2's reconstruction. Figure 5.5 presents the room seen during C-D course, so almost only rotating. Again, the alignment is not perfect but T5's results are better. Furthermore we can see that SACIA-ICP gave a better result than ICP.

We notice that in some cases, clouds having been inserted do not seem to be well aligned with the rest of clouds. For example, Figure 5.7 shows two clouds inserted in the map for test T2. The reason why the clouds of 5.7(a) are not aligned correctly is that the cloud was inserted without any alignment because the last 3 clouds had been rejected and its own alignment result was $0.80944 > 0.01$. This error is assumed



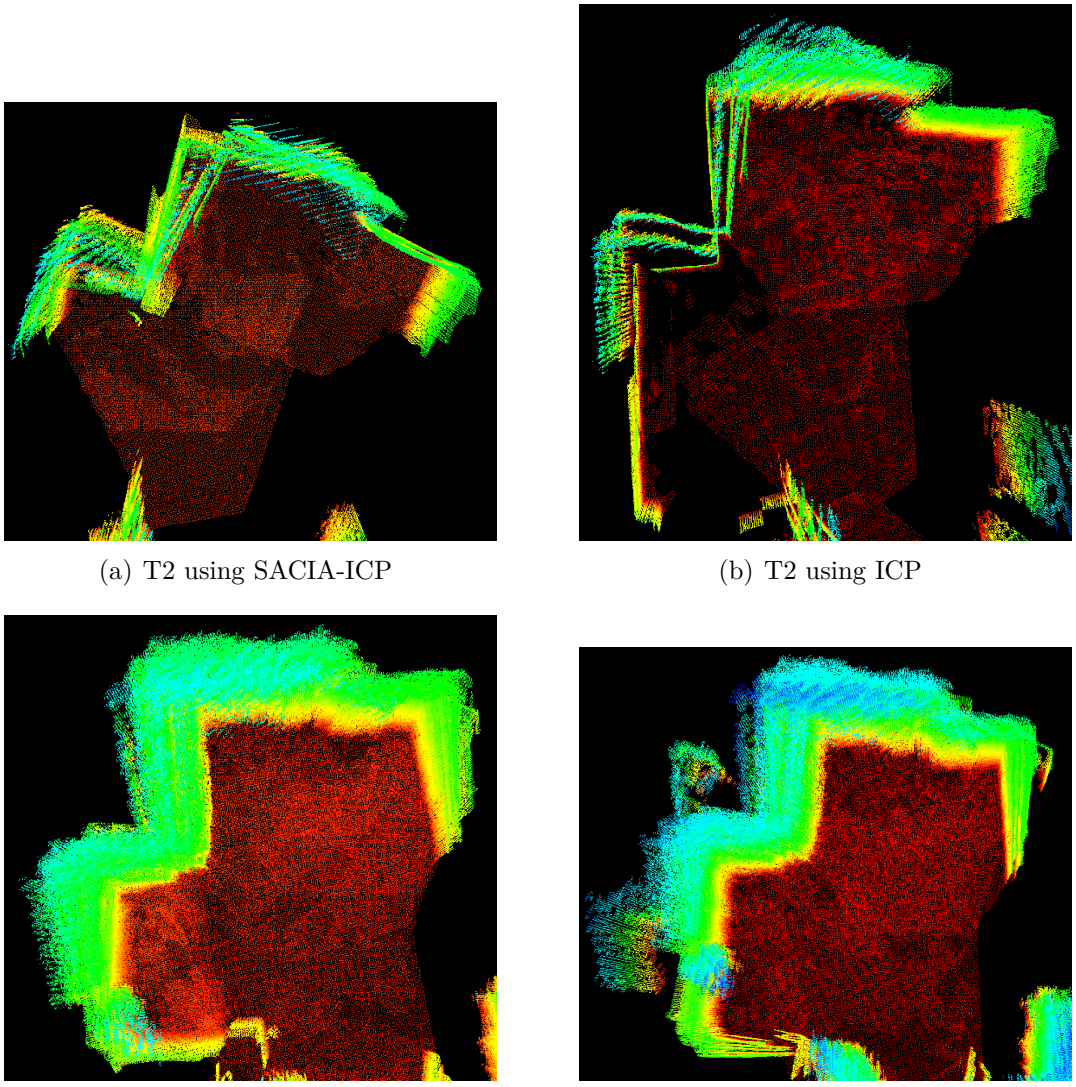
(a) T5 using SACIA-ICP



(b) T5 using ICP

Figure 5.4. Reconstructed maps obtained for test T5 using SACIA-ICP and ICP

in order to avoid a blockage of the registration process, where no new clouds would be accepted because there would be no overlapping area anymore. The clouds of 5.7(b) are correctly aligned, having a fitness score of 0.00014 after ICP alignment. We can see that the extinguisher and the fire hose are not blurry, meaning that even small objects can be reconstructed if the registration is good enough.



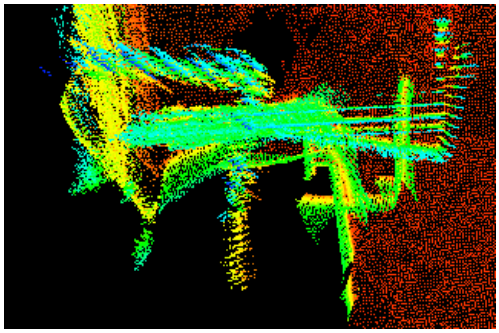
(a) T2 using SACIA-ICP

(b) T2 using ICP

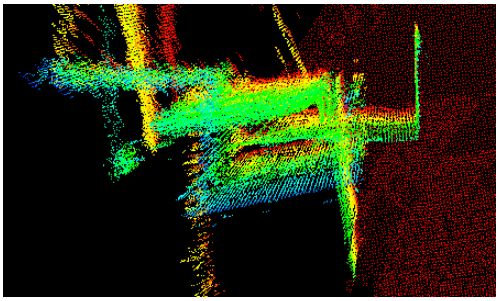
(c) T5 using SACIA-ICP

(d) T5 using ICP

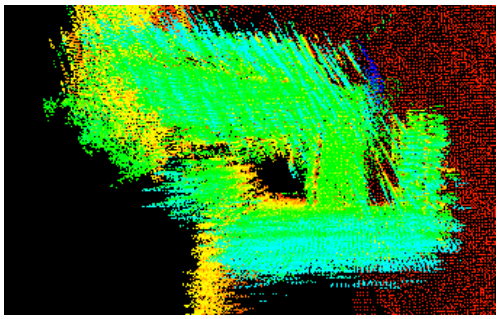
Figure 5.5. Details of a room seen while running C-D course for tests T2 and T5 using SACIA-ICP and ICP



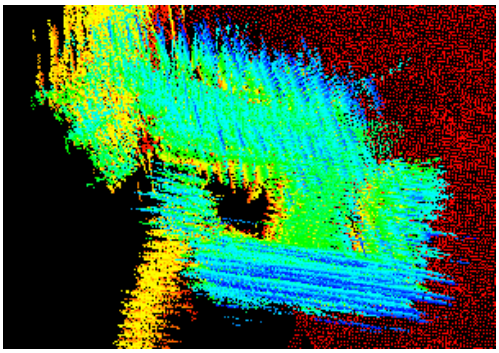
(a) T2 using SACIA-ICP



(b) T2 using ICP

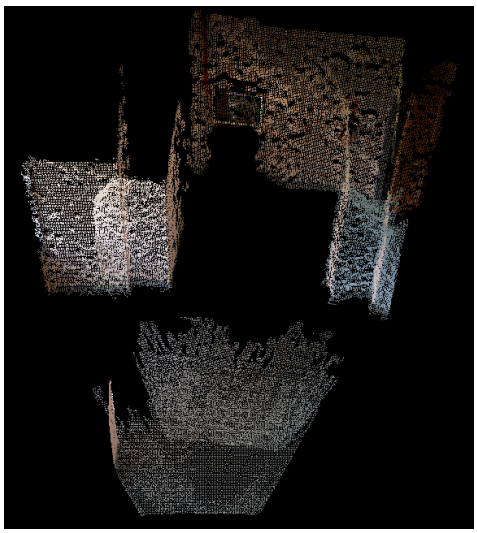


(c) T5 using SACIA-ICP

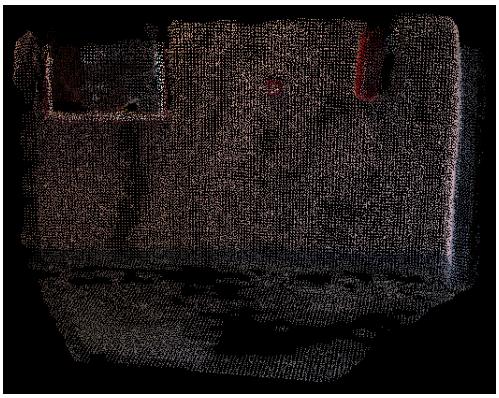


(d) T5 using ICP

Figure 5.6. Column details for tests T2 and T5 using SACIA-ICP and ICP



(a) Forced cloud

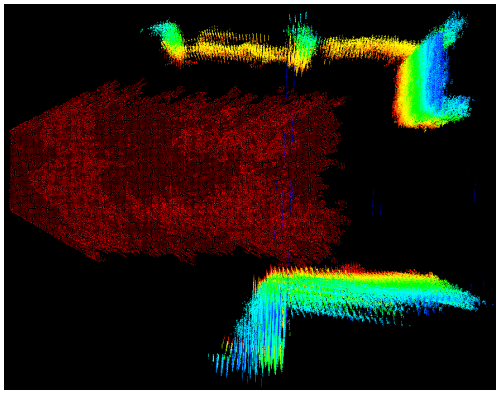


(b) Accepted cloud

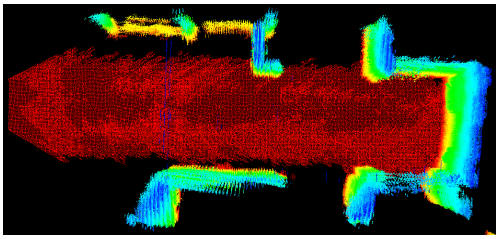
Figure 5.7. Clouds included in the map (a) after 3 previous clouds failed and (b) after registration process ended with a fitness score = 0.00014

5.5 Reconstruction sequence

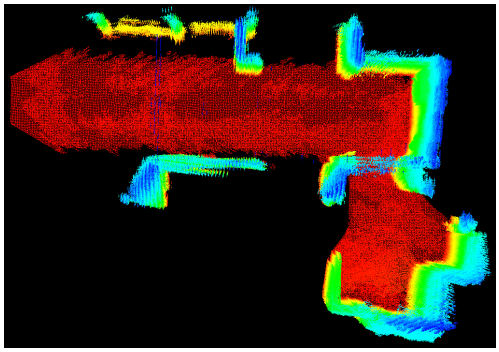
In order to have a better understanding of how the reconstruction process is made, we present in Figure 5.8 a sequence of the reconstruction made for T5 using SACIA and ICP alignment. Starting from position A (defined in section 5.1) 49 clouds have been accepted when 5.8(a) has been obtained and 145 clouds when 5.8(b). At this point, position B has been reached. 5.8(c) presents the reconstructed map when B-C has been covered and 259 clouds have been included. 5.8(d), 5.8(e) and 5.8(f) show the map corresponding to 259, 318 and 430 clouds accepted respectively, corresponding to travel C-D, D-B and B-A ways respectively. We observe that during the way back, so D-B and B-A, the reconstruction gets more inaccuracy as clouds of same objects but seen from a different point view do not succeed in aligning well.



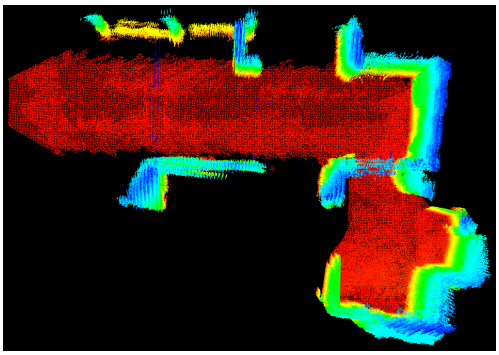
(a) 49 accepted clouds



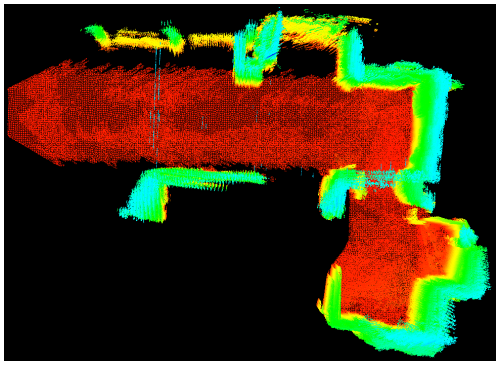
(b) 145 accepted clouds. A-B covered



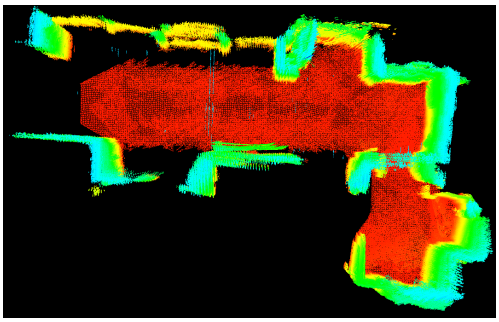
(c) 214 accepted clouds. B-C covered



(d) 259 accepted clouds. C-D covered



(e) 318 accepted clouds. D-B covered



(f) 430 accepted clouds. B-A covered

Figure 5.8. Sequence of maps obtained during T5 reconstruction

5.6 Results: Quantitative analysis

To realise a proper quantitative analysis, we understand that it would be convenient to find the distance error with respect to a 3D model of the environment. Though not having such 3D model, we will investigate on the percentage of clouds being accepted, rejected or forced, the latter being the most significant to understand the quality of it. We consider that alignments having a fitness score under 0.01 are acceptable. Such clouds included in the map will be understood as accepted clouds. Clouds not included in the map are defined as rejected while clouds included in the map because the three previous clouds during the registration process were rejected are considered as forced. We understand that forced clouds are source of maximal error as we cannot ensure an acceptable alignment but are needed in order to avoid a blockage of the reconstruction process. Tables 5.4 and 5.5 present the percentages obtained for every test using SACIA - ICP and ICP respectively as registration process.

	T1	T2	T3	T4	T5	T6	T7	T8
Accepted (%)	78.4	51.7	75.1	50.4	89.0	78.8	86.0	75.8
Rejected (%)	16.6	36.7	19.6	37.6	8.3	16.2	10.9	18.6
Forced (%)	5.0	11.7	5.3	12.0	2.7	5.0	3.1	5.5

Table 5.4. Percentages of clouds accepted, rejected and forced to be included for every test T1-T8 using SACIA and ICP

	T1	T2	T3	T4	T5	T6	T7	T8
Accepted (%)	74.3	36.7	65.3	39.3	86.1	67.2	83.6	67.4
Rejected (%)	19.9	48.3	26.7	46.2	10.6	24.9	13.1	25.0
Forced (%)	5.8	15.0	8.0	14.5	3.3	7.9	3.3	7.6

Table 5.5. Percentages of clouds accepted, rejected and forced to be inserted for every test T1-T8 using ICP only

A reconstructed map contains either accepted or forced clouds. Using only the clouds included in the map, we have analysed the percentage of forced clouds in order to understand better the influence of every factor in the quality of the resultant map. Such factors are the frame frequency f_{Acq}, the rover's speed v_{Rovmax}, the rover's rotational speed v_{Rotmax} and the registration process used.

To facilitate the notation we will refer to:

$$\begin{array}{lll} f_1 = 1 \text{ fps} & \text{and} & f_2 = 2 \text{ fps} \\ v_{Rov1} = 100 \text{ mm/s} & \text{and} & v_{Rov2} = 200 \text{ mm/s} \\ v_{Rot1} = 20 \text{ deg/s} & \text{and} & v_{Rot2} = 40 \text{ deg/s} \end{array}$$

$p_{Ti}(\%) \equiv$ percentage of forced clouds pertaining to a reconstructed map for test Ti.

5.6.1 Influence of the saving cloud frame frequency f_{Acq}

To analyse the influence of the saving frame rate we study the effect done between tests having the same parameters except for f_{Acq}. Furthermore, as we said in 5.3, in order to have a safer understanding of f_{Acq}'s influence, tests with f_1 have exactly the same data as their corresponding tests with f_2 but skip one cloud of two .

The corresponding tests are shown in Table 5.6 and are T1-T5, T2-T6, T3-T7, and T4-T8.

	v_{Rov1} v_{Rot1}	v_{Rov2} v_{Rot1}	v_{Rov1} v_{Rot2}	v_{Rov2} v_{Rot2}
f_1	T1	T2	T3	T4
f_2	T5	T6	T7	T8

Table 5.6. Corresponding tests for f_{Acq}

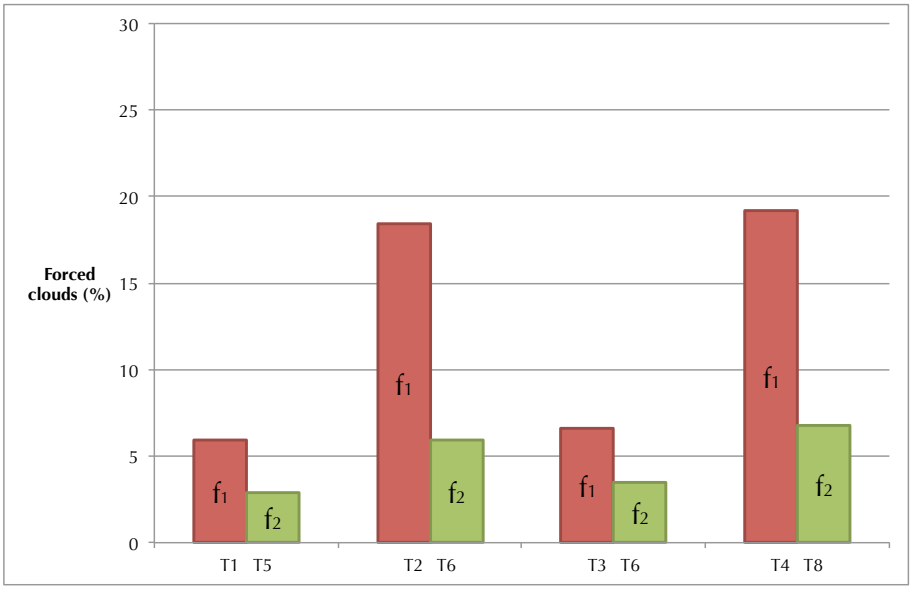
Figure 5.9 shows the percentage of forced clouds for f_{Acq} corresponding tests using both SACIA-ICP (Figure 5.9(a)) and ICP (Figure 5.9(b)) registration processes.

We notice that for every corresponding test, the one having f_2 brought less forced clouds than the one having f_1. Furthermore, we calculate in Table 5.7 both mean and standard deviation of the difference between forced cloud's percentage of f_{Acq} corresponding tests (for example, $p_{T1} - p_{T5}$).

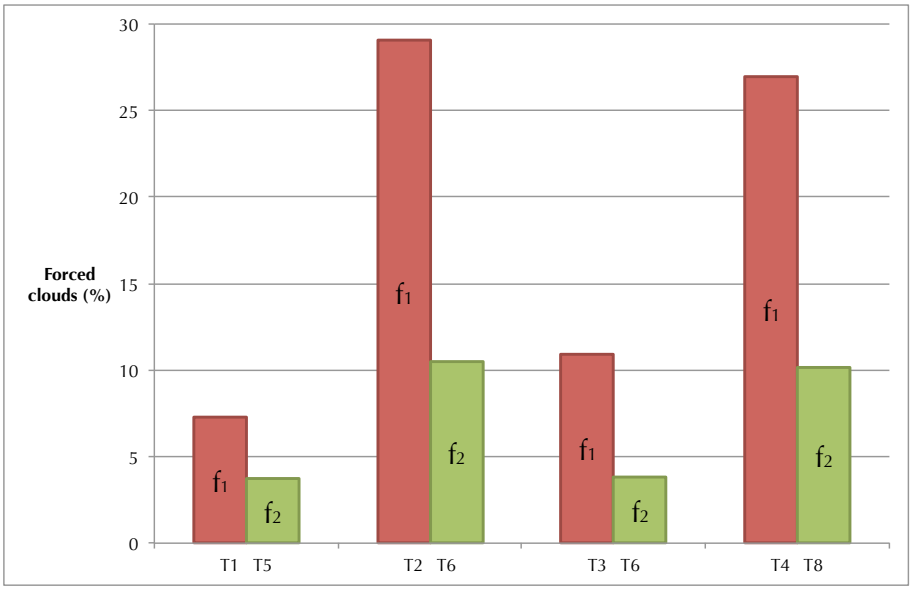
We can deduce that a faster acquisition of the clouds brings better results during the registration process.

	SACIA-ICP	ICP
Mean (%)	7.8	11.5
Standard deviation (%)	4.7	6.3

Table 5.7. Statistical results of the differences of p_{Ti} between f_{Acq}'s corresponding tests



(a) SACIA-ICP



(b) ICP

Figure 5.9. Influence of f_{Acq} in p_{Ti} for both (a) SACIA-ICP and (b) ICP registration process

5.6.2 Influence of rover's maximum rotational speed v_{Rotmax}

To analyse the influence of the rover's maximum rotational speed we study the effect done between tests having the same parameters except for v_{Rotmax}.

The corresponding tests are shown in Table 5.8 and are T1-T3, T2-T4, T5-T7, and T6-T8.

	v_{Rov1} f_1	v_{Rov2} f_1	v_{Rov1} f_2	v_{Rov2} f_2
v_{Rot1}	T1	T2	T5	T6
v_{Rot2}	T3	T4	T7	T8

Table 5.8. Corresponding tests for v_{Rotmax}

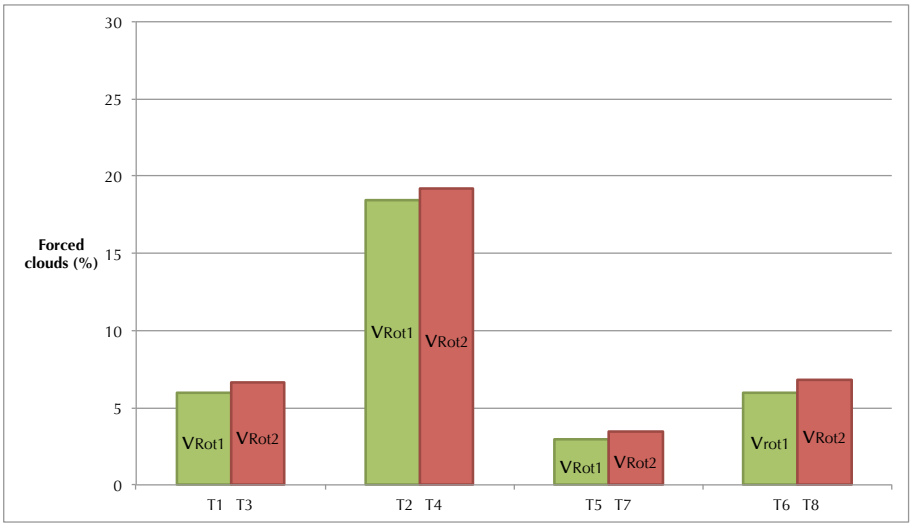
Figure 5.10 shows the percentage of forced clouds for v_{Rot} corresponding tests using both SACIA-ICP (Figure 5.10(a)) and ICP (Figure 5.10(b)) registration processes. We have also calculated in Table 5.9 both mean and standard deviation of the difference between forced cloud's percentage of corresponding tests (for example, $p_{T1} - p_{T3}$).

	SACIA-ICP	ICP
Mean (%)	-0.7	-0.4
Standard deviation (%)	0.1	2.1

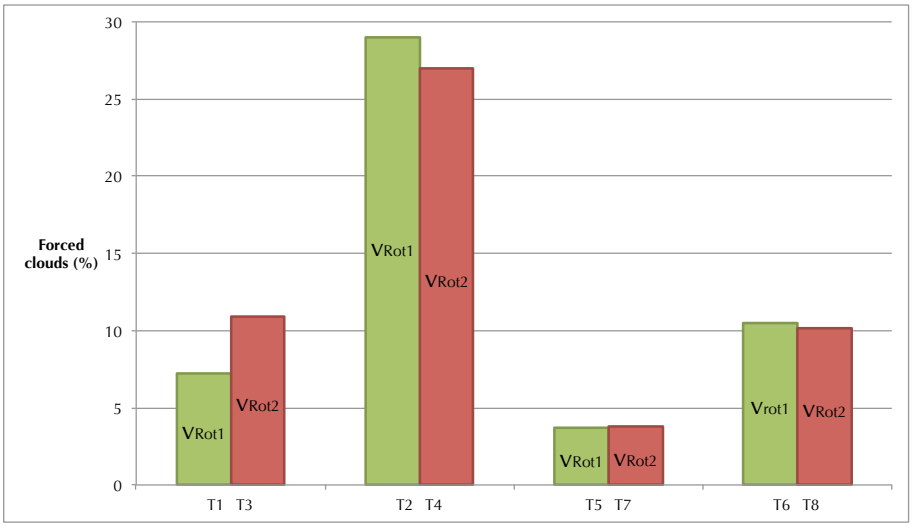
Table 5.9. Statistical results of the difference of p_{Ti} between v_{Rot}'s corresponding tests

We observe that increasing v_{Rotmax} has not always decreased the quality of the map as expected. Using SACIA-ICP, a little influence has been made, but using ICP, we cannot affirm the same as the standard deviation is much bigger than the mean and for T2-T4 and T6-T8, p_{Ti} decreased.

To deeply analyse these results, we have considered in table 5.10 the instantaneous rotational velocities v_{Roti} obtained while saving the pose during the tests. We observe that v_{Roti} is between 3 deg/s during the tests while the maximum rotational speed v_{Rotmax} is either 20 or 40 deg/s. Furthermore only 10.5 %, at the most, of the instantaneous speeds are greater than 10 deg/s. This means that we cannot conclude about the influence of rotational speed with this test as rotational speed does not significantly vary.



(a) SACIA-ICP



(b) ICP

Figure 5.10. Influence of v_{Rotmax} in p_{Ti} for both (a) SACIA-ICP and (b) ICP registration process

	T1	T2	T3	T4	T5	T6	T7	T8
Mean (deg/s)	1.3	2.3	2.1	3.4	1.3	2.3	2.0	3.6
Standard deviation (deg/s)	2.5	3.2	4.4	5.8	2.6	3.4	4.2	6.0
$v_{Roti} > 10$ deg/s (%)	2.9	5.0	4.9	10.1	2.7	0.4	4.9	10.5

Table 5.10. Statistical results of the instantaneous rotational velocity v_{Roti} and percentage of v_{Roti} having a value greater than 10 deg/s

5.6.3 Influence of rover's maximum speed v_{Rovmax}

To analyse the influence of the rover's maximum speed we study the effect done between tests having the same parameters except for v_{Rovmax}.

The corresponding tests are shown in Table 5.11 and are T1-T2, T3-T4, T5-T6, and T7-T8.

	v_{Rot1} f_1	v_{Rot2} f_1	v_{Rot1} f_2	v_{Rot2} f_2
v_{Rov1}	T1	T3	T5	T7
v_{Rov2}	T2	T4	T6	T8

Table 5.11. Corresponding tests for v_{Rovmax}

Figure 5.11 shows the percentage of forced clouds for v_{Rov} corresponding tests using both SACIA-ICP (Figure 5.11(a)) and ICP (Figure 5.11(b)) registration processes.

We notice that for every corresponding test, the one having v_{Rov1} brought less forced clouds than the one having v_{Rov2}. Furthermore, we calculate in Table 5.12 both mean and standard deviation of the difference between forced cloud's percentage of corresponding tests (for example, $p_{T1} - p_{T2}$). We can deduce that a slower motion behaviour of the robot brings better results during the registration process.

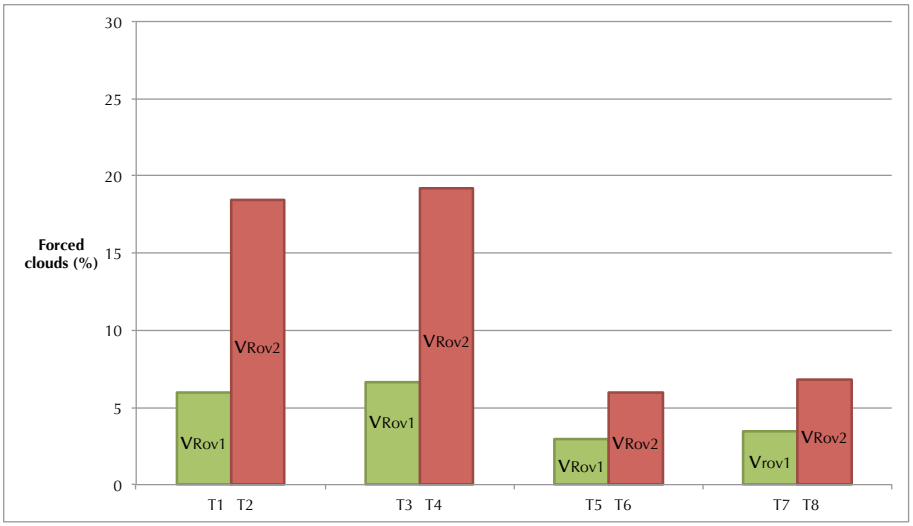
	SACIA-ICP	ICP
Mean (%)	-7.8	-12.7
Standard deviation (%)	4.7	6.5

Table 5.12. Statistical results of the difference of p_{Ti} between v_{Rov} corresponding tests

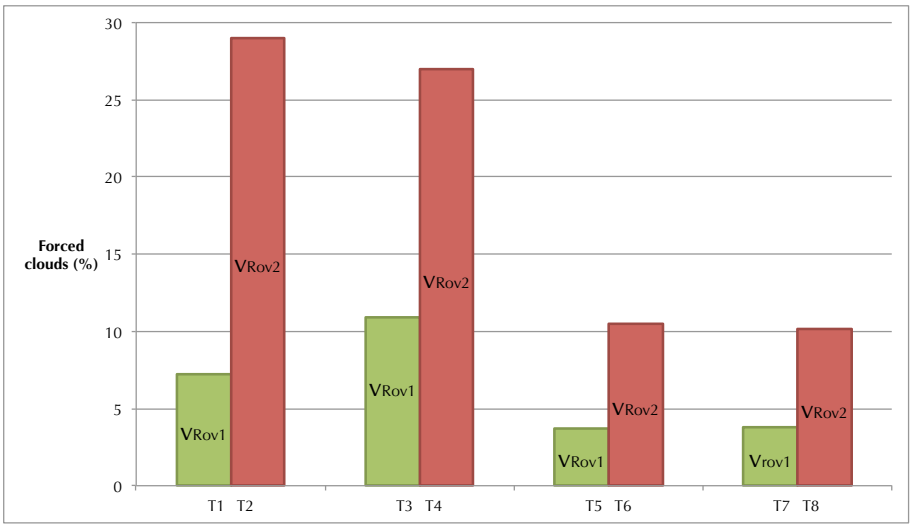
	T1	T2	T3	T4	T5	T6	T7	T8
Mean (mm/s)	69,7	129,6	74,2	133,8	69,8	128,8	74,5	134,1
Standard deviation (mm/s)	39,3	83,0	36,6	80,7	39,1	82,9	36,5	80,7
$v_{Rovi} > v_{Rovmax}/2$ (%)	63,6	58,7	69,0	60,5	63,6	58,3	69,0	61,2

Table 5.13. Statistical results of the instantaneous velocity v_{Rovi} and percentage of v_{Rovi} having a value greater than 50 mm/s or 100 mm/s, depending on if v_{Rovmax} is 100 mm/s or 200 mm/s respectively

To deeply analyse these results, we have considered in table 5.13 the rover's instantaneous velocities v_{Rovi} obtained while saving the pose during the tests. We observe that v_{Rovi}'s mean is of about 70 mm/s for tests having v_{Rov1} and 130 mm/s for tests having v_{Rov2}. Furthermore between 60 and 70 % of the instantaneous speeds are greater than their $v_{Rovmax}/2$. This means that the robot is sufficiently close to



(a) SACIA-ICP



(b) ICP

Figure 5.11. Influence of v_{Rovmax} in p_{Ti} for both (a) SACIA-ICP and (b) ICP registration process

its maximum speed during the test and so the conclusions made for v_{Rov} can be considered.

5.6.4 Influence of registration mode: SACIA-ICP vs ICP

To analyse the influence of registration process used, we study the results obtained by processing the tests with same parameters with both SACIA-ICP and ICP. As both aligning processes are done with the same data, conclusions can be safely done.

Figure 5.12 shows the percentage of forced clouds obtained p_{Ti} for the same tests processed using either SACIA-ICP or ICP.

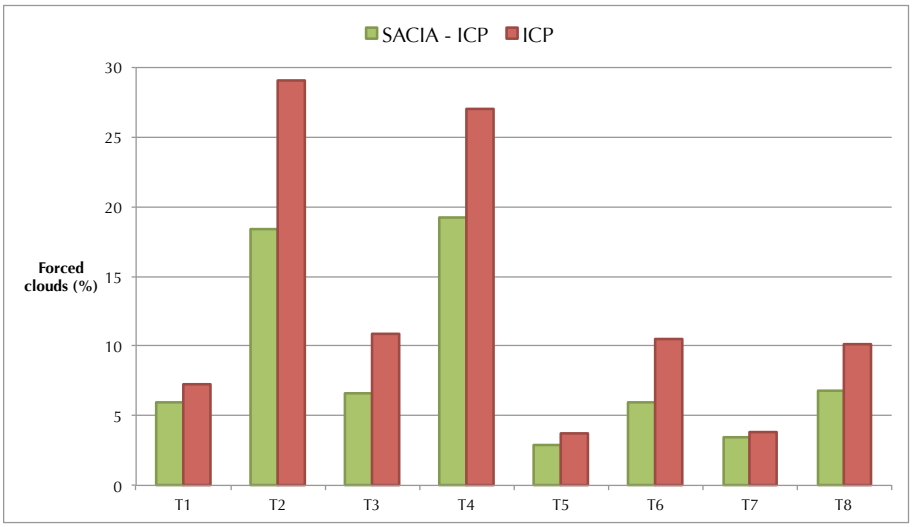


Figure 5.12. Influence of the registration process in the percentage of forced clouds obtained

We notice that for every test, registering using SACIA-ICP brings less forced clouds than aligning using only ICP. Furthermore, we have found that the difference between forced cloud's percentage of same tests (for example, p_{T1}(SACIA-ICP)$-p_{T1}$(ICP)) is of -4.1 % by mean and its standard deviation is 3.4 %. We can deduce that combining SACIA and ICP aligns better the acquired clouds than using only ICP.

5.7 Computing time analysis

We have seen that combining SACIA and ICP alignment processes, we can obtain maps of higher quality than registering with ICP only. However this improvement is not without cost. We have studied the computational time for both registration processes in order to analyse their cost. Table 5.14 presents the resultant mean $\bar{t}$ and standard deviation σ_t of the time spent. For SACIA-ICP, this time corresponds to process ICP, calculate new cloud's features if this one was included and, when required, calculate features, process SACIA and perform ICP again. On the other

hand, for ICP alignment the time used corresponds to ICP alignment only. Figure 5.13 shows the variation of $\bar{t}$ suffered by using a different alignment process. We can observe that the time spent using SACIA-ICP is greater than using ICP only as expected, and this augment is of 240 % by mean.

		T1	T2	T3	T4	T5	T6	T7	T8
ICP	$\bar{t}$ (s)	1,18	0,76	1,15	0,78	1,60	1,16	1,68	1,26
	σ_t (s)	0,61	0,56	0,80	0,58	0,80	0,73	1,01	1,03
SACIA-ICP	$\bar{t}$ (s)	3,22	4,32	4,11	4,47	2,89	3,20	3,78	3,35
	σ_t (s)	2,94	3,27	3,66	3,58	1,95	2,58	3,27	3,05

Table 5.14. Mean $\bar{t}$ and standard deviation σ_t of the computational time spent for the registration process using ICP or SACIA-ICP

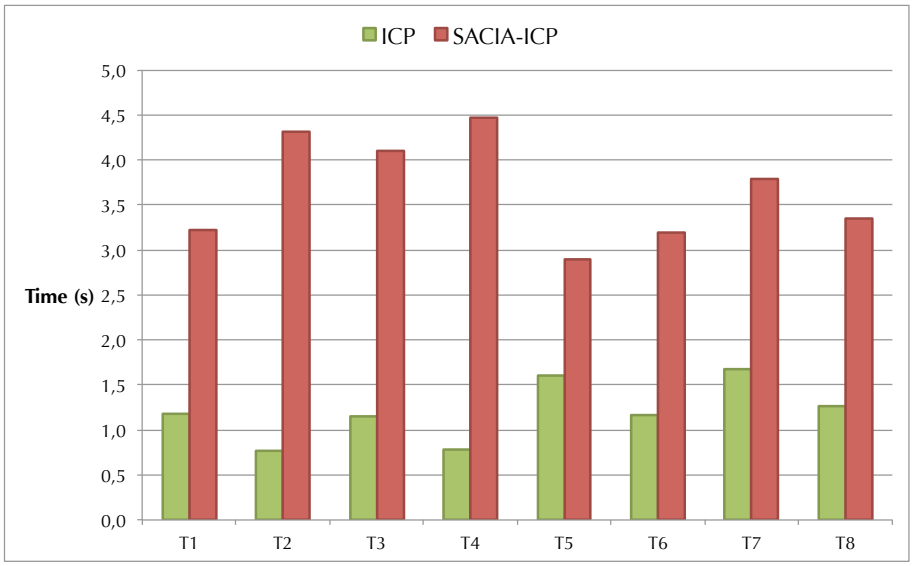


Figure 5.13. Influence of the registration process in the computational time

We observe though that the dispersion is considerable. To understand the reason of such variability, we observe in Figure 5.14 the time spent for each single alignment of test T5 using SACIA-ICP or ICP. We observe peaks when using SACIA-ICP that correspond to when SACIA alignment is done because ICP alignment has not achieve with an acceptable score. Such peaks augment considerably the dispersion as they suppose in some cases an augment of 400 % of the usual computational time.

Finally, we study in Figure 5.15 the evolution of the computational cost of Final process. As the map includes new clouds, its size grows, increasing the cost of doing

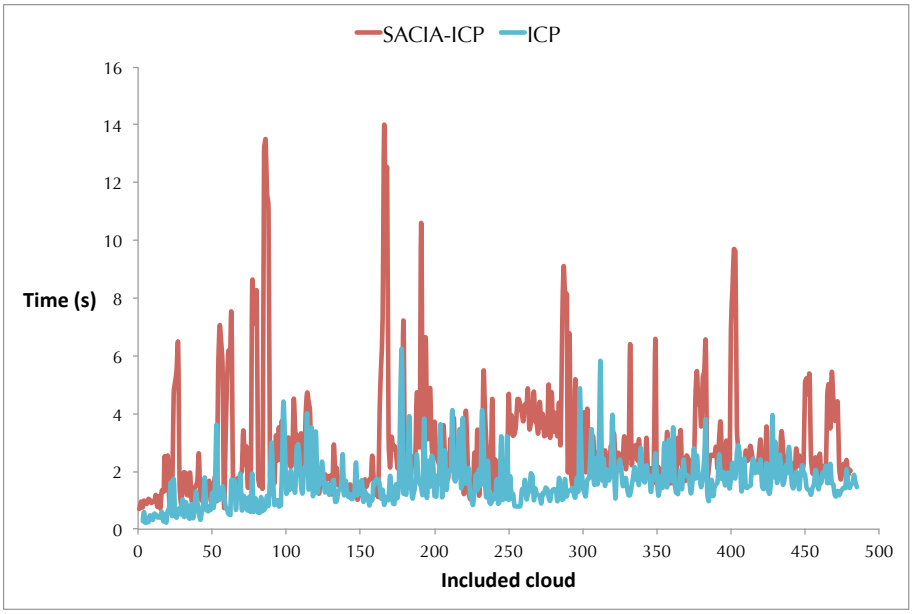
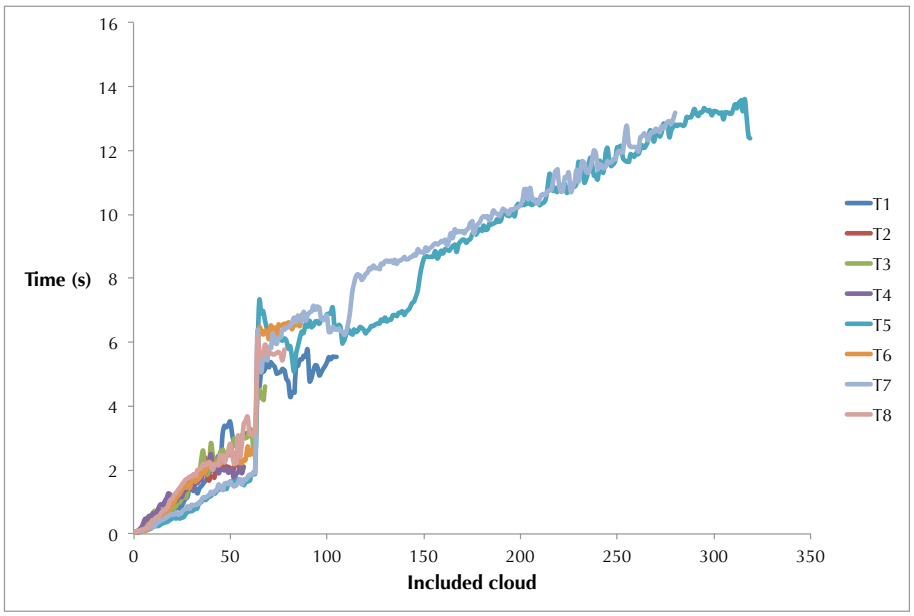
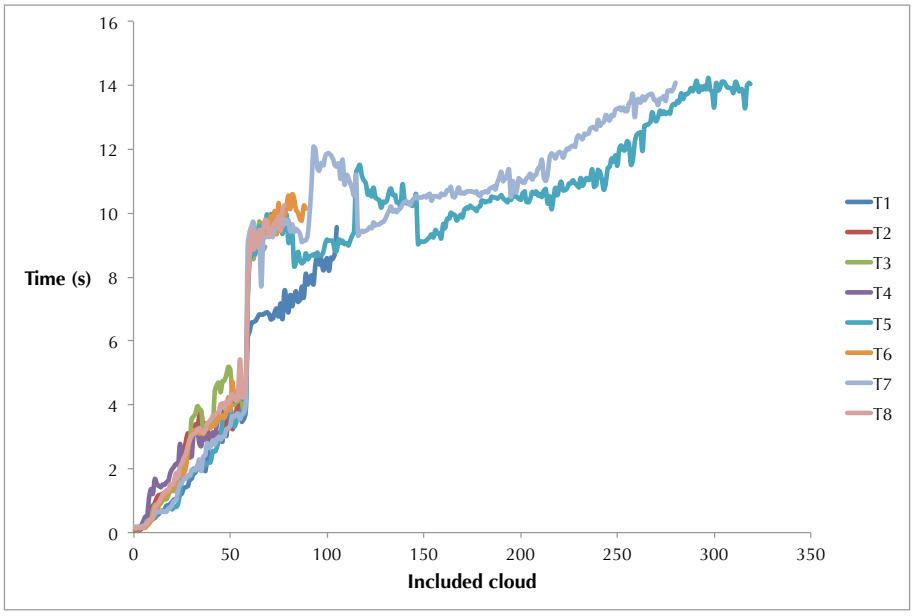


Figure 5.14. Time spent for every registration process of test T5

a Final process. This process is though necessary as if it were not done, the map's size would grow faster, increasing the registration's computational time. In order to maintain the computational cost of Final process bounded, it would be necessary to modify 3DLM in order to filter only the region where the new cloud was included.



(a) ICP



(b) SACIA-ICP

Figure 5.15. Evolution of the Final process' computational time for both ICP and SACIA-ICP registration processes

5.8 Conclusions

Dense reconstructions of indoor environments can provide to robotic systems new ways to navigate and interact with the surrounding world in an effective manner. The arise of inexpensive RGB-D cameras enables to build rich 3D maps that can be used for many applications such as navigation, manipulation or semantic mapping.

We introduced in this paper 3D Localised Mapping, a full algorithm that can reconstruct 3D indoor environments. To do so, it is necessary to provide to 3DLM the position and orientation of the camera. Furthermore, we have built two different variants on the registration processes being responsible for frame to frame reconstruction.

We have tested 3DLM for a particular environment, finding that for this particular case, having a faster acquisition speed or a slower movement of the rover has provided better 3D maps in a similar way. Furthermore, faster rotational speed has not had a strong influence on the quality of the resultant map as the rover does not turn as fast as expected due to the path defined. Finally, registration process combining SACIA and ICP has brought better results than using ICP only and appears to be so a better framework despite it requires greater computational times. Therefore, it is mandatory for a particular application to define its priorities in order to decide the registration process mode.

However, in order to have a better understanding of the robustness of the presented algorithm, more tests could be done, experimenting the influence of having more obstacles, being in larger or smaller environments, or changing registration parameters. A new test could be designed also to investigate on the influence of rotational movement. Furthermore, other issues remain to be tackled for future research, such as using RGB data in order to improve the registration process or optimizing the algorithm in order to obtain a real-time framework. To this purpose it would be mandatory to maintain all processes bounded to a convenient computational time. Moreover, as the rover remains in a plane environment, which is a common situation for indoor environments, it could be interesting to define a registration process that would find the best alignment using only x-y translation and z rotation. Finally, as the algorithm MMCL, used to localise the camera, performs better if several robots are operating together, 3DLM could be adapted to work with several rovers.

We hope that despite important issues need to be solved in order to improve 3DLM, this can be an opening step toward accurate 3D reconstructions.

Appendices

Appendix A

DataSheets

A.1 PrimeSense Reference Design 1.08

PrimeSense

The PrimeSensor™ Reference Design 1.08

NATURAL INTERACTION™

The PrimeSensor is an end-to-end solution that enables a computer to perceive the world in three-dimensions and to translate these perceptions into a synchronized image, in the same way that humans do. The solution includes a sensor component, which observes the scene (users and their surroundings), and a perception component, or *brain*, which comprehends the user interaction within these surroundings.

The PrimeSensor sees and tracks user movements within the scene and supplies the application layer with control widgets – thereby providing a simple clear API that translates user gestures or postures into known deterministic application inputs. All activity is performed without any assumptions about the host, the user or the environment. No wearable equipment is required, making the solution practical, convenient, intuitive and easy to use. The sensor provides a natural interface to living-room devices (such as game consoles and set top boxes), mobile devices and many more.

METHOD OF OPERATION

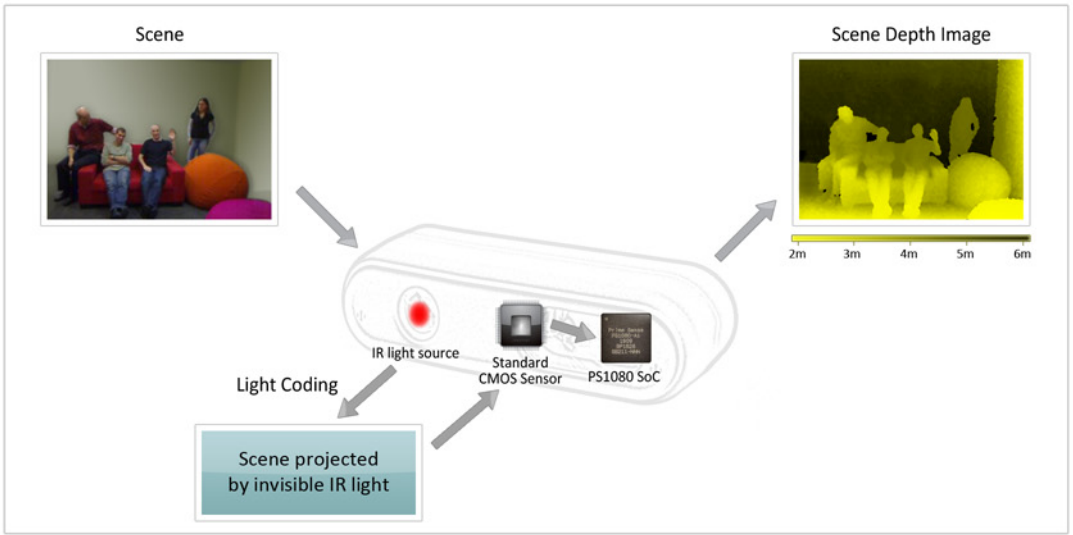
PrimeSense technology for acquiring the depth image is based on Light Coding™. Light Coding works by coding the scene volume with near-IR light. The IR Light Coding is invisible to the human eye. The solution then utilizes a standard off-the-shelf CMOS image sensor to read the coded light back from the scene. PrimeSense's SoC chip is connected to the CMOS image sensor, and executes a sophisticated parallel computational algorithm to decipher the received light coding and produce a depth image of the scene. The solution is immune to ambient light.

KEY BENEFITS

- Thin host natural interface
- Mass consumer market product price point
- Mass production proven solution
- Simple and clean application layer APIs
- Support a unique end-to-end solution – no need for the application layer to have depth processing capabilities

HIGHLIGHTS

- A large VGA-sized depth image
- Multiple sensing capabilities: depth image, color (RGB) image and audio stream
- Standard off-the-shelf components
- A unique *Registration* mechanism matches every depth image pixel to its true color image pixel
- A low-power embedded device – USB powered
- IEC 60825-1 Class 1 laser product



Founded in 2005, PrimeSense is an established fabless semiconductor startup company, and a clear leader in the field of sensory inputs for the high-volume consumer market. PrimeSense has achieved a series of patented technological breakthroughs in the field of three-dimensional machine vision that provide consumer devices with realtime, per-frame full-motion three-dimensional capturing and processing abilities. PrimeSense is the only provider of such capabilities at mass consumer market prices, and the sole provider of a holistic end-to-end sensor input solution.

PrimeSense | The PrimeSensor™ Reference Design 1.08

TECHNICAL OVERVIEW

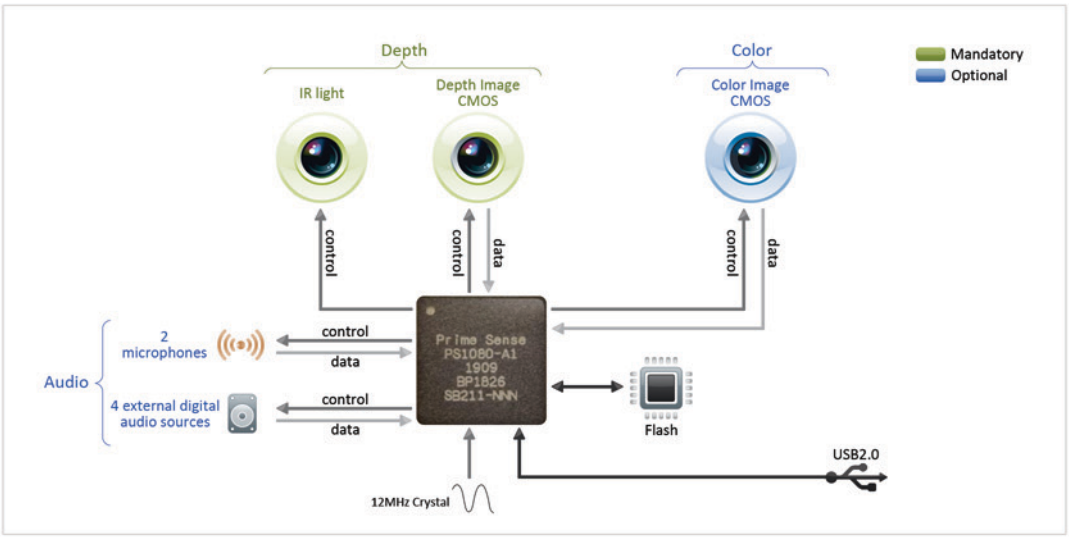
The PrimeSensor is built around PrimeSense's PS1080 SoC. The PS1080 controls the IR light source in order to project the scene with an IR Light Coding image. The IR projector is a Class 1 safe light source, and is compliant with the IEC 60825-1 standard. A standard CMOS image sensor, receives the projected IR light and transfers the IR Light Coding image to the PS1080. The PS1080 processes the IR image and produces an accurate per-frame depth image of the scene.

The PrimeSensor includes two optional sensory input capabilities: color (RGB) image and audio (the PrimeSensor has two microphones and an interface to four external digital audio sources).

To produce more accurate sensory information, the PrimeSensor performs a process called *Registration*. The Registration process's resulting images are pixel-aligned,which means that every pixel in the color image is aligned to a pixel in the depth image.

All sensory information (depth image, color image and audio) is transferred to the host via a USB2.0 interface, with complete timing alignment.

BLOCK DIAGRAM



PRODUCT SPECIFICATION

Property	PrimeSensor Spec
Field of View (Horizontal, Vertical, Diagonal)	58° H, 40° V, 70° D
Depth image size	VGA (640x480)
Spatial x/y resolution (@2m distance from sensor)	3mm
Depth z resolution (@2m distance from sensor)	1cm
Maximal image throughput (frame rate)	60fps
Average image latency in full VGA resolution	40msec
Operation range	0.8m - 3.5m

Property	PrimeSensor Spec
Color image size	UXGA (1600x1200)
Audio: built-in microphones	2 mics
Audio: digital inputs	4 inputs
Data interface	USB 2.0
Power supply	USB 2.0
Power consumption	2.25W
Dimensions (Width x Height x Depth)	14cm x 3.5cm x 5cm
Operation environment (every lighting condition)	indoor
Operating temperature	0°C - 40°C

DELIVERABLES

- Reference Design 1.08 Evaluation Kit:
  - PrimeSensor
  - Gesture API demo
  - Application demo (media center, casual games, web browsing, text entry)
- Reference Design 1.08:
  - Schematics
  - Full BOM list
  - Datasheets
  - PCB layout
  - Mechanical design
  - Acquisition firmware (PS1080 firmware)
  - Host driver
  - Device Development Kit (DDK)
- Manufacturing:
  - Reference Design knowledge transfer from PrimeSense to customer, enabling customer to build own sensor, or
  - Customer engages with PrimeSense's ODM procuring assembled units
- World-class support
  - Hardware and software support teams
  - Close support during hardware and software design and bring-up phases
  - On-site training

PrimeSense Israel
28 Habarzel St.
Tel-Aviv, 69710, ISRAEL
Tel: +972.3.769.2222

PrimeSense N. America
555 Bryant St. Suite 578
Palo Alto, CA 94301, USA
Tel: +1.650.862.5089

PrimeSense Taiwan
2F., No.4, Aly. 24, Ln. 91,
Sec. 1, Neihu Rd., Neihu Dist.,
Taipei City 114, Taiwan (R.O.C.)
Tel: +886.918.538.861

PrimeSense Japan
Mr. Mitsuo Ukaji in e-Connections KK
Yokohama Creation Square 13F
5-1 Sakaechyo, kanagawa-ku,
Yokohama-shi 221-0052, JAPAN
Tel: +81.45.461.1999

A.2 MobileRobots Pioneer 3DX

Pioneers

are the world's most popular intelligent mobile robots for education and research. Their versatility, reliability and durability have made them the reference platforms for advanced intelligent robotics. Unlike hobby and kit robots, Pioneers are fully programmable, as bundled with Pioneer SDK, and will last through years of tough classroom and laboratory use.

Pioneer 3-DX

is a small, lightweight, two-wheel, two-motor differential drive robot, ideal for indoor laboratory and classroom use.

All Pioneers come complete with wheel encoders and a mobility microcontroller with ARCOS firmware, one lead-acid battery expandable to three hot-swappable, Pioneer SDK advanced mobile robotics development software and lifetime access to updates and upgrades. Pioneers also provide user-accessible 12 and 5 VDC power, as well as digital and serial I/O for customization.

All Pioneers also support a plethora of fully integrated accessories, including SONAR, gripper, 6-DOF arm, laser-range finder with advanced localization and navigation software, onboard SBC, wireless Ethernet, mono and stereo vision, gyro/IMU, automated charging, bumpers, and many more…

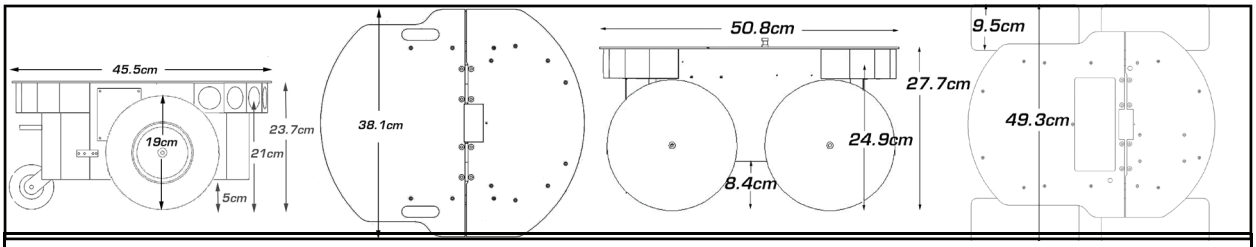
Pioneer 3-AT

is a small, four-wheel, four-motor skid-steer intelligent robotics platform ideal for heavier payloads and all-terrain operation on unconsolidated surfaces (indoor wheels available for operation on carpets and other high-friction surfaces).

Additional accessories include outdoor stereo vision, differential GPS, and more…

Adept MobileRobots, LLC • 10 Columbia Drive • Amherst, New Hampshire 03031
http://www.MobileRobots.com • sales@MobileRobots.com
+1 603 881-7968 • +1 603 881-3818 fax

Pioneer 3-DX and –AT Specifications



		Pioneer 3-DX	Pioneer 3-AT
Physical Characteristics*		14 guage aluminum body	and chassis
	Length (cm)	48.5	50.8
	Width (cm)	38.1	49.3
	Height (cm)	23.7	27.7
	Clearance (cm)	5.0	8.4
	Weight (kg)	9	14
	Payload (kg)	25	40
Power		12VDC unregulated; 12 and 5VDC	regulated switched user auxiliary
	Batteries	Up to 3 hot-swappable	12VDC@7.2 A-hr ea
	Run-time (hrs)	8-10 ; continuous with hot-swap	4-6 ; continuous with hot-swap
	Run:charge time ratio	0.9:1	0.9:1
Mobility	Tires	2 foam-filled@19.5cm diam, 4.8cm wide; 7.5cm rear caster	4 pneumatic@22cm diam, 9.5cm wide; knobby tread
	Turn:Swing radii (cm)	0:26.7	0:34
	Translate speed max (mm/sec)	1,400	700
	Rotate speed max (deg/sec)	300	140
	Traversable step:gap max (cm)	2.0:8.9	8.9:12.7
	Traversable slope (grade)	25%	40%
	Terrain	Wheelchair accessible indoors	Unconsolidated *(no carpets!)*
Microcontroller		44.2368 MHz Renesas SH2 32-bit FLASH; ARCOS firmware	RISC µC with 32KRAM and 128K
Encoders (2 quadrature)		313,657 ticks per rev	135,465 ticks per rev
SONAR		Front and rear (optional) arrays of	8 multiplexed, 180-degrees @ 20-
		degree intervals each	
Switches and buttons		Main power, AUX1 and AUX2 power,	µC reset, and MOTORS
Ports		SYSTEM serial, charger,	and joystick
User I/O		SYSTEM serial and 8 digital inputs	and outputs
Systems Architecture		Server in client-server; Pioneer SDK includes Linux and Windows® ARIA ware, MobileSim simulator, SonARNL Basic and MobileEyes™ configuration	software bundled with every robot, C++ robotics client development soft- localization and navigation, Mapper3- and control GUI application.

* Based on standard configurations with single battery; no accessories. DX includes front SONAR array.

Adept MobileRobots, LLC • 10 Columbia Drive • Amherst, New Hampshire 03031
http://www.MobileRobots.com • sales@MobileRobots.com

A.3 Laser Rangefinder SICK LMS-200

12 Technical Data

Type	**Indoor: LMS200, Outdoor: LMS211, LMS221, LMS291**
Scanning angle (field of vision)	100°/180° (type-dependent, see *Table 11-1, Page 27*)
Motor speed	75 Hz
Angular resolution (response time)	0.25°[1] (53.33 ms); 0.5° (26.66 ms); 1° (13.33 ms); selectable
Range	Max. 80 m (type-dependent, see *Table 11-2, Page 27*)
Measurement resolution	10 mm
Measurement accuracy	typical ±35 mm (LMS200-30106, LMS211/LMS221-30106: typical ±15 mm)
Systematic error[2]	LMS200-30106/LMS211-30106/LMS221-30106: – mm-mode: typical ±15 mm at range 1 to 8 m – cm-mode: typical ±4 cm at range 1 to 20 m LMS211/LMS211-30206/LMS291/LMS221-30106/LMS2x1-Sxx: – mm-mode: typical ±35 mm at range 1 to 20 m – cm-mode: typical ±5 cm at range 1 to 20 m
Statistical error[3]	LMS200-30106/LMS211-30106/LMS221-30106: mm-mode: typical 5 mm at range ≤ 8 m/ reflectivity ≥ 10 %/ light ≤ 5 klx LMS211/LMS211-30206/LMS291/LMS221-30106/LMS2x1-Sxx: mm-mode: typical 10 mm at range 1 to 20 m/ reflectivity ≥ 10 %/ light ≤ 5 klx
Laser diode (wavelength)	Infra-red (λ = 905 nm)
MTBF of LMS2xx[4]	Indoor devices: 70,000 h Outdoor devices: 50,000 h
Laser class of device	Class 1 (eye-safe), to EN/IEC 60825-1 and to 21CFR 1040.10
Optical indicators	3 x LED (LMS200/LMS291 only)
Data interface	RS 232 or RS 422 (selectable in the connector plug)
Data transfer rate	RS 232: 9.6 / 19.2 kbd RS 422: 9.6 / 19.2/ 38.4/ 500 kbd
Data format	1 start bit, 8 data bits, 1 stop bit, no parity (fixed)
Switching inputs	All LMS2xx except LMS2xx-S14 (LMS Fast): 1 x (“Restart“ or “Field set switching“), U_{in} = 12 to 24 V, I_{in} = 5 mA
Switching outputs (standard device)	LMS200/LMS291/LMS211/LMS221: 3 x PNP (OUT A to OUT C), high, typical 24 V DC (“field OK“), short-circuit-proof, selectable restart delay after field infringement (0; 100 ms to 255 s) – OUT A, OUT B (each max. 250 mA): “field infringement“ – OUT C (max. 100 mA): “field infringement/error indication (Weak)[5]“

1) Angular resolution 0.25° not possible in the area monitoring mode

2) Definition measurement accurcy:
Resolution:
The resolution of a measuring device is the smallest possible distance different from zero between two consecutive individual measurement values.
The resolution can be reduced by using averaged values.
Systematic error:
Environmental conditions: good visibilty, T_a = 23 °C, reflectivity 10 to 10,000 %.
The systematic error is the sum of all the deviations over a defined excent of range and reflectivity, which cannot be reduced even using averaged values.

3) **Statistical error:**
Standard deviation 1 sigma. The standard deviation is calculated using at least 100 measuring values of a target (object) with a certain reflectivity at a certain distance with a certain amount of illumination.

4) at T_a = 25 °C

5) For indicating an error, the output pulses with 1 Hz/5 Hz and different pulse ratios depending on the error type, see *Table 7-4, Page 21*.
Special devices LMS211/221-S19/-S20: indication of front window contamination (warning/error) additionally via OUT C switching output by statical signal

Table 12-1: Technical specifications of LMS200/LMS211/LMS221/LMS291

Appendix B

Parameters used for 3D reconstruction of a corridor

B.1 Filtering process

The parameters used for all filtering processes are shown in table B.1

B.2 Features calculation

The parameters used for features estimation are shown in table B.2

B.3 SAmple Consensus Initial Alignment

The parameters used for SACIA are shown in table B.3

B.3.1 Iterative Closest Points

The parameters used for ICP are shown in table B.4

B.3.2 Decision and Final Process

The parameters used for ICP are shown in table B.4

Passthrough Filter	Limits (m)	[0; 6]
	Filter Filed Name	z
Down Sampling	Size Voxel Grid (m)	0.02
Remove Outliers	Number of points k	50
	Standard Deviation Multiplier Threshold	1
Surface reconstruction	Radius search	0.6

Table B.1. Parameters used for Filtering process

Down Sampling	Size Voxel Grid (m)	0.06
Remove Outliers	Number of points k	50
	Standard Deviation Multiplier Threshold	1
Normal estimation	Radius search	0.2
FPFH estimation	Radius search	0.3

Table B.2. Parameters used for calculating Features

Maximum Correspondence Distance (m^2)	$0.3 \cdot 0.3$
Minimum Sample Distance (m)	0.4
Maximum number of iterations	1000
Transformation epsilon	$1e^{-5}$

Table B.3. Parameters used for SACIA

Maximum Correspondence Distance (m^2)	$0.15 \cdot 0.15$
Maximum number of iterations	500
Transformation epsilon	$1e^{-5}$

Table B.4. Parameters used for ICP

Decide	Minimum acceptable Fitness Score	0.01
Final process	Size Voxel Grid (m)	0.6

Table B.5. Parameters used for Decision and Final process

Bibliography

[ABI+08] Fabrizio Abrate, Basilio Bona, Marina Indri, Stefano Rosa, and Federico Tibaldi. Switching Multirobot Collaborative Localization in Symmetrical Environments. In *IROS 2008 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, pages 94–99, 2008.

[AIR+09] Fabrizio Abrate, Marina Indri, Stefano Rosa, Federico Tibaldi, and Basilio Bona. Three-State Multirobot Collaborative Localization in Symmetrical Environments. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 1–6, 2009.

[BM92] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[HKH+10] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[KA08] Kurt Konolige and Motilal Agrawal. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[Kon10] Kurt Konolige. Projected Texture Stereo. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[MDH+09] Stefan May, David Droeschel, Dirk Holz, Stefan Fuchs, Ezio Malis, Andreas Nüchter, and Joachim Hertzberg. Three-Dimensional Mapping with Time-Of-Flight Cameras. *Journal of Field Robotics, Special Issue on Three-Dimensional Mapping, Part 2*, 26(11-12):934–965, December 2009.

[Mes] Mesa imaging. http://www.mesa-imaging.ch/.

[NÖ9] Andreas Nüchter. *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom.*, volume 52 of *Springer Tracts in Advanced Robotics*. Springer, 2009.

[NSS+09] Paul Newman, Gabe Sibley, Mike Smith, Mark Cummins, Alastair Harrison, Chris Mei, Ingmar Posner, Robbie Shade, Derik Schröeter, Liz

Murphy, Winston Churchill, Dave Cole, and Ian Reid. Navigating, Recognizing and Describing Urban Spaces With Vision and Lasers. *The International Journal of Robotics Research*, 28(11-12), 2009.

[Pri] PrimeSense. http://www.primesense.com/.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[Rus09] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muechen, Germany, October 2009.

[Zha92] Zhengyou Zhang. Iterative Point Matching for Registration of Free-Form Curves. Research Report RR-1658, INRIA, 1992.