Non rigid transformation based Localization and Mapping

Group 51

170387F – M. W. G. Vihan Melaka

Contribution – *Full report and all the references*

Supervisors

Dr. Chandana Gamage
Dr. Sulochana Sooriyaarachchi

9th December 2021

2017 Intake

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa

Table of Content

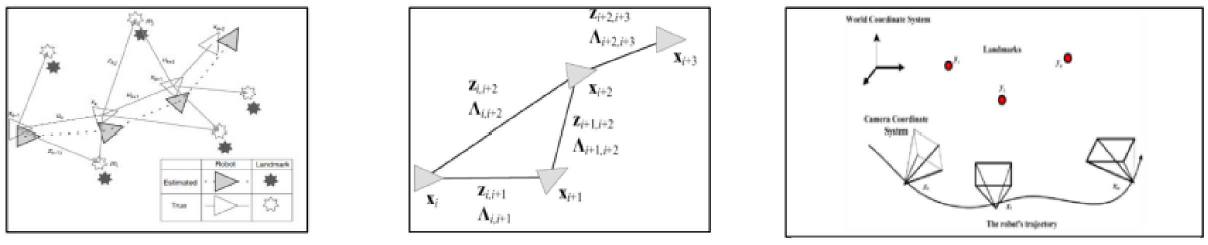
List of Figures

1. Introduction

1.1 Background

With the emergence of autonomous vehicles and improvement in the robotics field, the need of having a proper way to localize the robot of vehicles in the environment and mapping of the environment was an undeniable requirement of the field. Localization of a robot means knowing exactly where the robot is at the moment in the environment in order to achieve this robot must have some knowledge about the environment that is where the requirement of the map of the environment comes into action. Over the years' scientists and researchers tried to tackle localization problem and from the results they got they got to know that localization problem is a non-converging problem if it was to solve as an independent problem. But when considering to solve localization problem together with mapping at the same time localization problem was a converging problem. This was the beginning of the Simultaneous Localization and Mapping or SLAM concept in the robotics field.

Over the years many SLAM representations emerged like EKF-SLAM which is based on extended Kalman filter and probabilistic estimation of the landmarks in the map, graph SLAM which used a graph to keep track of the environment and recently more advanced SLAM techniques which uses data from the RGB-D cameras, LIDAR's were came into the field and SLAM approaches based on these were collectively called as visual SLAM, SLAM methods like ORB-SLAM, RGB-D SLAM, Elastic fusion falls under this category. Although there are many solutions to this problem, every technique has its pros and cons and they are suitable for a specific scenario and conditions. Because of solutions being specific to a certain scenario there is still space for development of new SLAM techniques that will benefit an unexplored set of conditions or improvement of an existing solution to adapt more requirements is also possible.



a) Probabilistic SLAM b) Graph SLAM c) Visual SLAM

Figure 1.1 – SLAM representations

State of the art solution for visual SLAM problem consists of two parts called the front end and the backend [1]. Frontend is where all the data from the sensors is processed and localization of the robot is happening using the available data in the environment. Backend is the part where loop closures are handled. Loop closure is identifying the same location in the environment and marking that on the map as a single location. Neglecting loop closure part from the solution will simply reduce the SLAM problem to odometry. Without a loop closure map will be an infinite corridor but with the introduction of loop closure robots are able to identify loops in this corridor and they are being able to find shortcuts from one point to another without following the same path which it used to get to that point. Therefore, both parts of the SLAM which is frontend and backend is equally important when creating a solution to the SLAM problem.

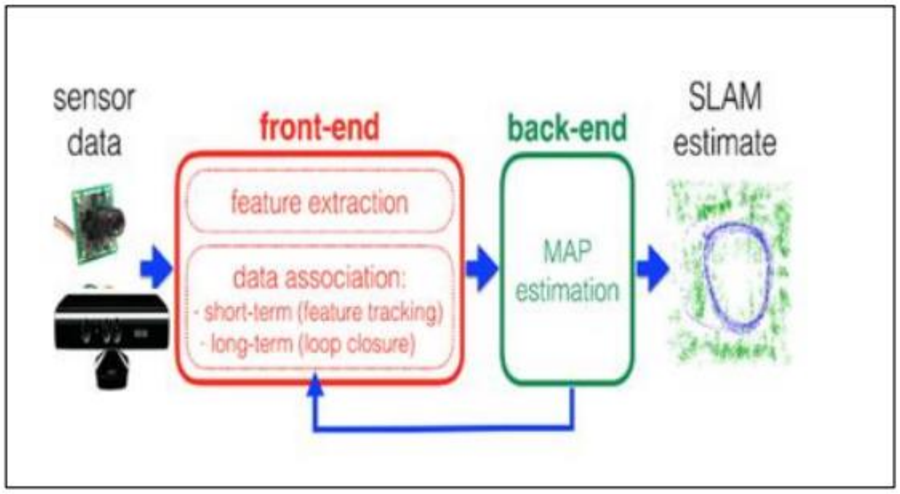


Figure 1.2 Typical model of Visual SLAM

This project is an attempt to implement a new visual SLAM approach which uses deformation theory at the backend of the SLAM system which was proposed in the Elastic fusion SLAM approach proposed by T. Whelan et al. Deformation theory involves deforming the map in a loop closure scenario, which will cause individual points in the map to be move relative to each other which causes a *Non rigid transformation* in the map.

1.2 Problem Statement

As we know there are two important parts in the state of the art solution for visual SLAM as frontend and the backend. For the frontend different SLAM approaches use different types of input data and different techniques of landmark extraction but in the backend there is only a limited number of techniques to use. One of the most important tasks in the backend is to reduce the error that has been accumulated in the mapping process. Most of the SLAM approaches use a technique called pose graph optimization which keeps a graph of frames captured and pose of the camera at that moment as nodes and edges representing the connection between two nodes, by maintaining the graph pose graph optimization technique keeps track of the camera trajectory. In a loop closure situation when the backend receives two frames with identical features normally there is a small error in the pose associated with the same features, features tend to be a bit out of position. In pose graph optimization technique this error is handled by distributing accumulated error across the pose graph as shown in Figure 1.3. Issue with pose graph optimization technique is that because it has to maintain a pose graph, memory consumption is high, therefore it is unable to continue to map a large environment.

In a recent paper called Elastic Fusion [14] they proposed a method of using deformation theory [11] which is used in the video animation field to deform shapes at the backend of the SLAM system. In this method deformation theory was used to deform the map of the environment in a loop closure scenario to reduce the accumulated error as shown in Figure 1.4. In the proposed method they used surfels to build the map of the environment and surfel features was used to build the deformation graph which was extracted from the RGB-D images captured from a Kinect sensor. Because of maintaining a single map and deforming it when necessary, memory consumption in the Elastic fusion approach is less when compared to pose graph optimization approach.

At present there are many visual SLAM approaches and most of them uses pose graph optimization at the backend [9][3][8]. The only SLAM approach that uses deformation theory concept at the backend is the Elastic fusion [14] SLAM approach. To explore more capabilities of using concept of deformation theory at the backend of visual SLAM systems, in this project a new SLAM approach is proposed which uses deformation theory at the backend by using 3D point clouds to build the map of the environment and using point cloud features to build the deformation graph. Frontend of the SLAM system will be using robots odometry data along with point clouds when building the map.

"A new SLAM approach using deformation theory at the backend and using 3D point clouds to build the map of the environment"

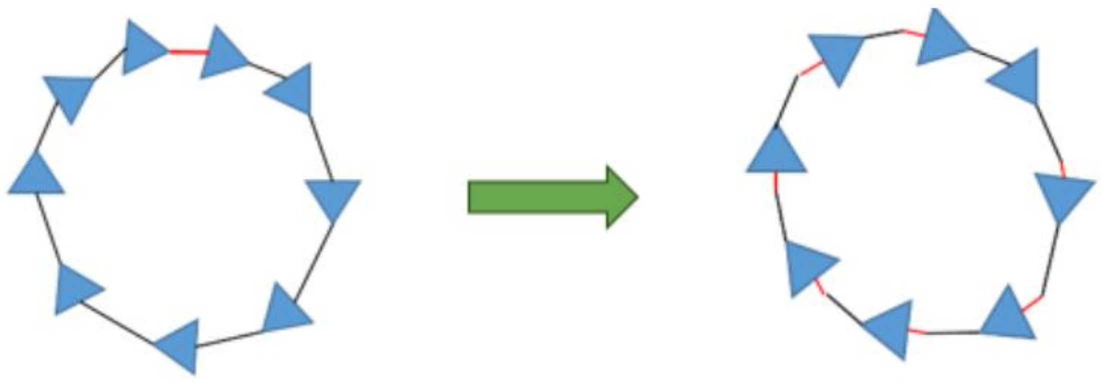


Figure 1.3 How error at the loop closure is handle in pose graph optimization technique

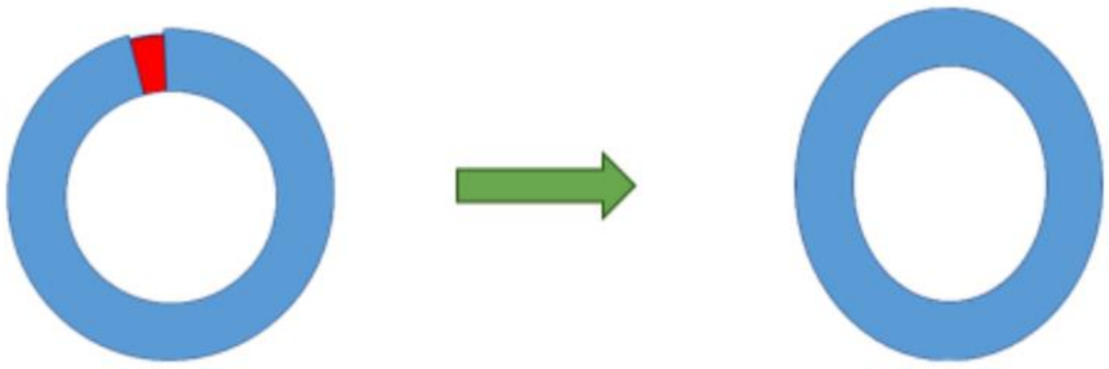


Figure 1.4 How error at the loop closure is handle in Elastic fusion using deformation theory

1.3 Research Objectives

Following measures will be taken to achieve the goal of the project which is to build a SLAM system based on the work of Whelan et al [14].

- At the frontend a new data type will be introduced to the SLAM approach after examining the compatibility of the data type when using with deformation theory.
- A research will be conducted on appropriate features to use in order to carry on pose estimation based on the input data type.
- Deformation theory [9] will be tested on new input data types and how to construct the deformation graph using features of new data types will also be analyzed.
- Propagating deformations from points in the deformation graph to other points that exist in the map will be carried on a GPU-based environment using CUDA.

At the end of the implementation, the proposed solution will be tested on a simulated environment built using gazebo.

2. Literature Review

2.1 Introduction

The most important question of why SLAM is important in the robotics field is answered in the introduction section of the work by Cadena et al on Past, Present and Future of Simultaneous Localization and Mapping. According to the work, SLAM in the robotics field is estimating the state of a robot equipped with on board sensors, and the construction of a model (the map) of the environment that the sensors are perceiving [1]. Furthermore, the paper shows that SLAM is not useful, for instance in an outdoor environment we can use GPS (the GPS satellites can be considered as beacons at known locations) to localize the robot and navigate in the environment. According to the work of David et al, the popularity of the SLAM problem is connected with the emergence of indoor applications of mobile robotics.

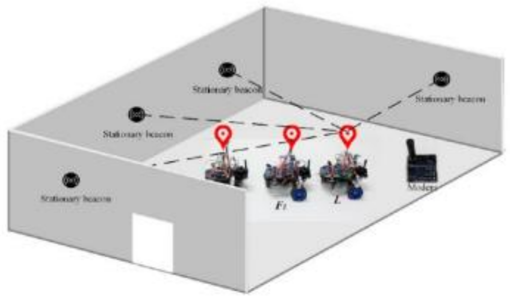


Figure 2.1 Using beacons for Localization

Figure 2.2 Using GPS for Localization

2.2 History of SLAM

In "The history of the SLAM problem" section of The Journal paper by H. Durrant and T. Bailey on SLAM essential algorithms [2] gives us a good overview of the initialization and evolution of the SLAM problem over the years. According to the paper at the 1986 IEEE Robotics and Automation Conference held in San Francisco first discussion about the probabilistic SLAM was discussed and as a result it was recognized that consistent probabilistic mapping was a fundamental problem in robotics with major conceptual and computational issues that need to be addressed [2]. The paper discussed different approaches taken by researchers to solve the SLAM problem and problems faced by them, especially how the SLAM community got to know that the combined mapping and localization problem, once formulated as a single estimation problem, was actually convergent [2]. From there onwards the paper gives basic ideas on probabilistic SLAM and the theory and concepts behind the probabilistic SLAM approach.

Work by Cadena et al divides the evolution of SLAM problem into two phases as classical age (1986-2004), algorithmic-analysis age (2004-2015) [1]. According to the paper, classical age was the primitive stage of the SLAM problems which includes main probabilistic formulations for SLAM such as approaches based on Extended Kalman Filters, Blackwellized Particle Filters and Maximum likelihood estimation. The algorithmic-analysis age was centered around fundamental properties of SLAM, including observability, convergence and consistency. Furthermore, work of Cadena et al explains what makes SLAM unique from odometry by focusing on loop closure concept and introduces modern state of the art SLAM system which consists of frontend and a backend.

At present most of the SLAM approaches fall into the category of visual SLAM. From the work of A. Huletski et al on evaluation of the modern visual SLAM methods [7] we can see that most of the modern SLAM approaches use vision frontend to capture data from the environment. The reason for the emergence of visual SLAM is because of the expensiveness of other sensors like LIDAR's using a monocular camera or a RGB-D camera for gathering data is much more efficient.

2.3 Related Work

Project is mainly built on work of Whelan et al about the Elastic fusion SLAM approach [14]. In this paper they are integrating deformation theory which is used in the video editing field to manipulate shapes to deform a surface in order to construct a map of the environment. This is a more map centric approach to SLAM meaning it mainly focuses on building a detailed map than having precise pose estimations. Most significant feature of this approach is how it handles loop closures using deformation theory. This implementation also uses GPU based implementation for both pose estimation and in the creation of the map.

Other than deformation theory [11] used in Elastic fusion [14] there are concepts like pose graph optimization and bundle fusion which are being used in loop closure instances in the robotics field. Related work section of the paper by R. Mur-Artal and J. D. Tardos on ORB-SLAM2 [9] we can see that most of the RGB-D SLAM approaches use pose graph optimization in their backend. According to the paper Kintinous by Whelan et al [13], RGB-D SLAM by Endres et al. [3] and DVO-SLAM by Kerl et al [8] SLAM approaches use pose graph optimization in the backend for loop closure. But in the work of S. Rahman et al [8] on simulation of SLAM using 3D point cloud data, in the section 2.2 about Graph-SLAM they state that one of the disadvantages of SLAM that uses pose graph is that it requires high memory computations as it incorporates all the pose estimates during the calculation process which is an issue when mapping larger environment. When compared to pose, the graph optimization method used in Elastic fusion is more memory efficient as it only maintains a single map instead of multiple frames associated with pose.

Another method that is used in the work of R. Mur-Artal and J. D. Tardos [9] is bundle fusion which is a technique used in constructing 3D environments. In the work they state that their goal is long-term and globally consistent localization instead of building the most detailed dense reconstruction which is complete opposite of the goal of work by Whelan et al. Elastic fusion by Whelan et al places much more emphasis on the accuracy of the reconstructed map over the estimated trajectory [14].

However, in the related work section of the work of R. Mur-Artal and J. D. Tardos on Orb-slam2 [7], they mention about the Elastic fusion approach stating that the detailed reconstruction and localization accuracy is impressive but the problem is that the current implementation is limited to room-size maps as the complexity scales with the number of surfers in the map which is the motivation for this project. In this project a new SLAM approach which is based on work of Whelan et al on Elastic Fusion is proposed with a new data type instead of surfels which will reduce the complexity of the computation.

2.4 Deformation Theory

Deformation theory is a shape deformation technique that is mostly used in video graphics field. For the deformation of the shape this theory uses a reduce model of the shape called "Deformation graph". Then the reduce model is deformed using affine transformations [9] [10]. Finally, those transformations which were applied to the reduce model is applied to the other points in the shape in order to create the final deformed shape. Basically this process consists of four main steps.

1) Extracting key points from the initial shape to create the reduce model.
2) Building the reduce model or the deformation graph using the extracted the key points.
3) Apply affine transformations to the points in the reduce model to deform the reduce model to get the desired deformation.
4) Apply the deformations that was applied to the key points in the reduce model to the points in the shape which are not key points in order to get the final deformed shape.

This process is shown in Figure 2.3 using a simple example. And how this theory has used in Elastic Fusion SLAM approach is shown in the Figure 2.4. Top part of the image shows how deformation graph is built when an environment is mapped in a motion from left to right and then right to left and bottom part of the image shows how graph is getting built with the time.

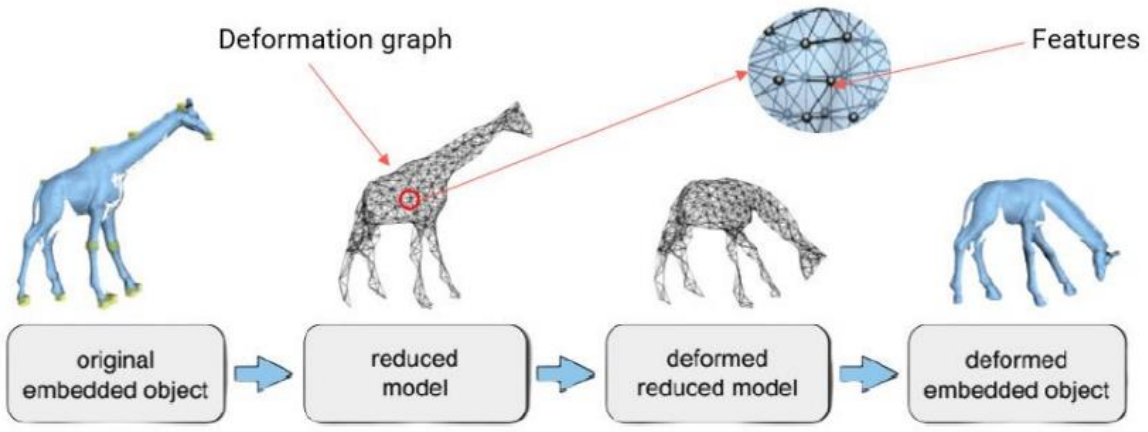


Figure 2.3 – Steps of deformation theory [11]

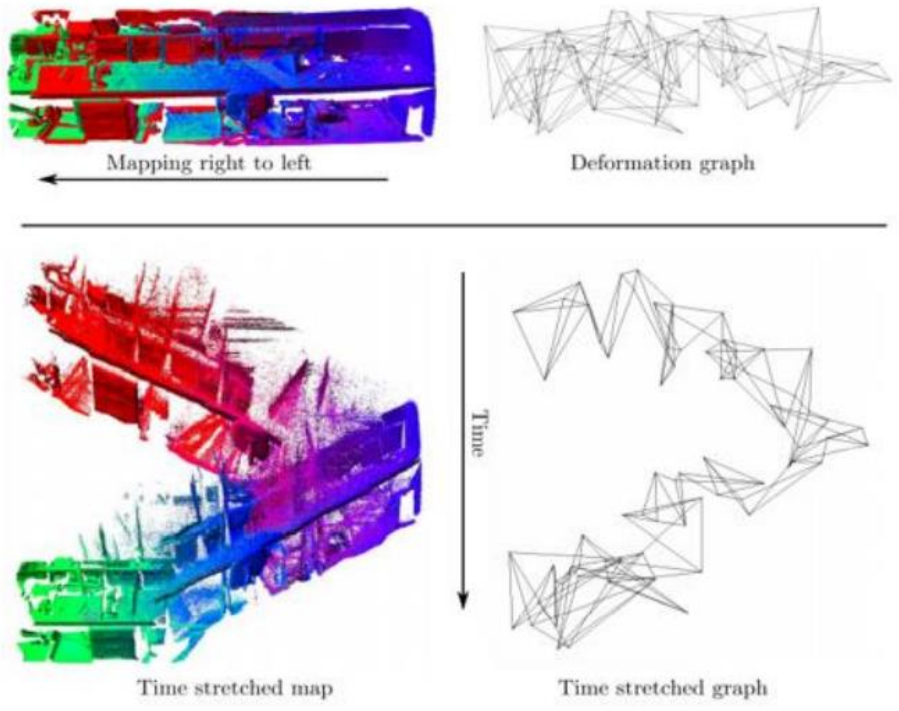


Figure 2.4 – Use of deformation theory in Elastic fusion [14]

3. Methodology

In the proposed SLAM method point cloud data is proposed to be used as the input data type for the system. Method of deformation theory [11] will be used in the backend for the loop closure. In the proposed approach a traditional and computationally cheap approach based on point clouds such as iterative closest point or feature based for localization will be used. Steps to follow in this type of a point cloud registration approach and some suggestions about the features that can be used is indicated in the work by Gongora Velandia [4] on Mapping of indoor environments using point cloud library.

When making the deformation graph for the mapped environment Elastic fusion [14] used most impactful surfels as the nodes of the graph, in this implementation deformation graph is proposed to be built using point cloud features. When distributing transformations to other points which are not a part of the deformation graph, Elastic Fusion uses GPU based approach and in the proposed solution the same approach will be used. Overall proposed methodology is indicated as a workflow diagram in Figure 3.1

This project is planned to be carried out in a simulation environment using Gazebo and a simulated robot that runs ROS as the operating system. When working with point clouds, the PCL library will be used to manipulate and process data. When working with GPUs Nvidia CUDA is used as in the work of Whelan et al. [14].

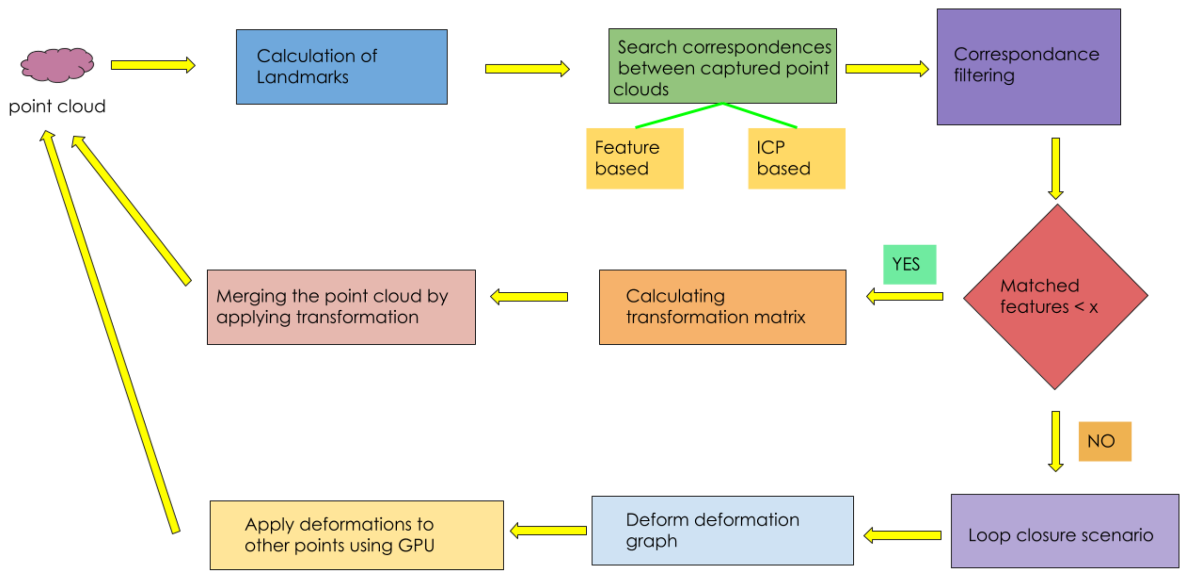

point cloud
Calculation of Landmarks
Search correspondences between captured point clouds
Feature based
ICP based
Correspondance filtering
Matched features < x
YES
NO
Calculating transformation matrix
Merging the point cloud by applying transformation
Loop closure scenario
Deform deformation graph
Apply deformations to other points using GPU

Figure 3.1 - Proposed methodology for SLAM, threshold value of “x” is yet to be calculated in an experimental manner.

4. Experimental Evaluation

According to the proposed methodology the first step of the implementation was the calculation of landmarks.

4.1 Calculation of Landmarks (key points)

Calculation of landmarks or key points in the environment was an important step in this implementation because it is important to have a consistent key points extraction method when calculating the error in a loop closure scenario. Since point cloud data is used as the input data type, both 2D and 3D key point extraction algorithms can be used. When using 2D key point extraction algorithms it is important to use the same frame that is given by point cloud by converting into a 2D image for key point extraction rather than using an input from a different input source because the coordinates given by the key point extraction algorithm will depend on the focus and the resolution of the input source. After performing a research on the existing key point extraction algorithms following algorithms were chosen to perform an experimental research to select the best performing algorithm to move into the next steps.

Selected 2D key point extraction algorithms are (using OpenCV library),

- Harris corner detector
- SIFT (Scale Invariant Feature Transform)
- SURF (Speeded Up Robust Features)
- FAST (Features from Accelerated Segment Test)
- ORB (Oriented FAST and rotated Brief)

Selected 3D key point extraction algorithms are (using PCL library),

- ISS (Intrinsic Shape Signature)
- AGAST (Improved version of FAST)
- SIFT (Scale Invariant Feature Transform)
- NARF (Normal Aligned Radial Feature)
- Brisk (Binary Robust Invariant Scalable Key points)
- Susan (Uses intensity of the surrounding points to calculate key points)
- Trajkovic (Uses intensity of the surrounding points to calculate key points)

Performance analysis of two algorithm sets were done individually in the same testing environment and selected the best performing algorithm out of two sets and those two algorithms were compared to choose the best algorithm to calculate key points in the implementation. When analyzing the performance of the algorithms speed of the algorithm, consistency of calculating key points was considered as the benchmarks. As the testing environment two objects with different surface natures was used, i.e. A box shaped garbage can and a traffic cone as shown in Figure 4.1.

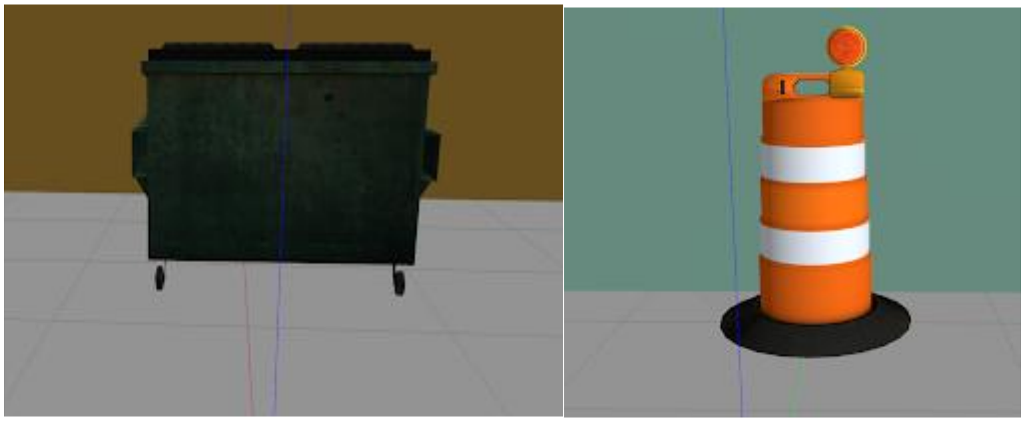


Figure 4.1 – Experimenting environment for testing key point extraction algorithms

First 2D key point extraction algorithms were tested in the experiment and the results were as follows, (Note that Harris corner detector algorithm is not tested in the experiment because after conducting a research it was found that algorithm is not scalar invariant.)

The specifications of the computer and the resolution of the point cloud used in the analyzing task is as follows,

- Resolution – 1080 x 1920 pixels
- RAM – 4GB
- Processor – Intel Core i5

1. Garbage box scene

Algorithm	**Execution time**	**Key points detected**	**Consistency**
SIFT	2.5 seconds	130 – 150	Satisfactory
SURF	1.15 seconds	680 – 700	Inconsistent
ORB	0.15 seconds	400 – 500	Satisfactory
FAST	0.1 seconds	160 – 180	Inconsistent

2. Traffic cone scene

Algorithm	Execution time	Key points detected	Consistency
SIFT	2.5 seconds	50 – 70	Satisfactory
SURF	1.2 seconds	290 – 330	Inconsistent
ORB	0.14 seconds	180 – 200	Satisfactory
FAST	0.1 seconds	90 – 120	Inconsistent

By considering the results, it is concluded that ORB algorithm performs better than other algorithms when considering the time consumed and the consistency of the key points captured. Therefore, ORB algorithm is the selected key point extraction algorithm that will be compared with the best performing 3D key point extraction algorithm.

Then 3D key point extraction algorithms were tested in the experiment and the results were as follows, (Note that ISS algorithm is not tested in the experiment because after conducting a research and an experiment it was found that algorithm is not scalar invariant [12].). This experiment was done in two steps because some of the algorithms was not available in the PCL 1.8 version that was being used and had to switch working stations to test the other key point extraction algorithms. First the SIFT, AGAST, NARF, Susan algorithms were tested with PCL 1.8 with a computer with the following specifications,

- Processor – Intel Core i5
- RAM – 4GB

And a point cloud with 1080 x 1920 pixels' resolutions.

1. Garbage box scene

Algorithm	**Execution time**	**Key points detected**	**Consistency**
SIFT	2.2 seconds	140 – 160	Inconsistent
AGAST	0.15 seconds	60 – 70	Satisfactory
NARF	2.8 seconds	40 – 50	Neutral
Susan	60 seconds >	30 – 40	Inconsistent

2. Traffic cone scene

Algorithm	**Execution time**	**Key points detected**	**Consistency**
SIFT	1.38 seconds	50 – 70	Inconsistent
AGAST	0.25 seconds	50 – 60	Satisfactory
NARF	2.9 seconds	30 – 40	Neutral
Susan	60 seconds >	0	-

From the results, it was observed that AGAST key point extraction algorithm had the best performance. Then the work station was changed and rebuilt the ROS working space in the new working station and had to change the simulated robot that was using because older robot did not support the new version of the ROS. Therefore, the resolution of the point cloud was also changed from 1080 x 1920 to 480 x 640. The specifications of the new working station are as follows,

- Processor – Intel Core i7
- RAM – 4GB

Results of the analysis of the remaining 3d key point extraction algorithms with the best performing algorithm from previous experiment is as follows,

1. Garbage box scene

Algorithm	Execution time	Key points detected	Consistency
AGAST	0.01 seconds	60-70	Satisfactory
BRISK	0.01 seconds	30 – 35	Neutral
Trajkovic	1.1 seconds	-	-

2. Traffic cone scene

Algorithm	Execution time	Key points detected	Consistency
AGAST	0.01 seconds	50 – 60	Satisfactory
BRISK	0.01 seconds	10 – 15	Neutral
Trajkovic	1.6 seconds	-	-

After analyzing all the selected 3D key point extraction algorithms, AGAST algorithm was selected as the best performing 3D key point extraction algorithm in terms of execution time as well as consistency of capturing key points.

As the final step of selecting a suitable key point extraction algorithm, Analysis between the best performed 2D key point extraction algorithm (ORB) and best performed 3D key point extraction algorithm (AGAST) was done by performing the same experiment and results are as follows,

1. Garbage box scene

Algorithm	Execution time	Key points detected	Consistency
ORB	0.01 seconds	300-400	Satisfactory
AGAST	0.01 seconds	60 – 70	Satisfactory

2. Traffic cone scene

Algorithm	Execution time	Key points detected	Consistency
ORB	0.01 seconds	180 – 200	Satisfactory
AGAST	0.01 seconds	50 – 65	Satisfactory

When analyzing the results, it was clear that both algorithms were performing equally in terms of time of execution and the consistency of the key point extraction, but ORB algorithm was able to capture more key points than the AGAST algorithm. Therefore***, ORB 2D key point extraction algorithm*** was chosen as the key point extraction algorithm to move into the next steps. The results obtained from the two best performing algorithms in two scenarios is shown in the Figure 4.2.

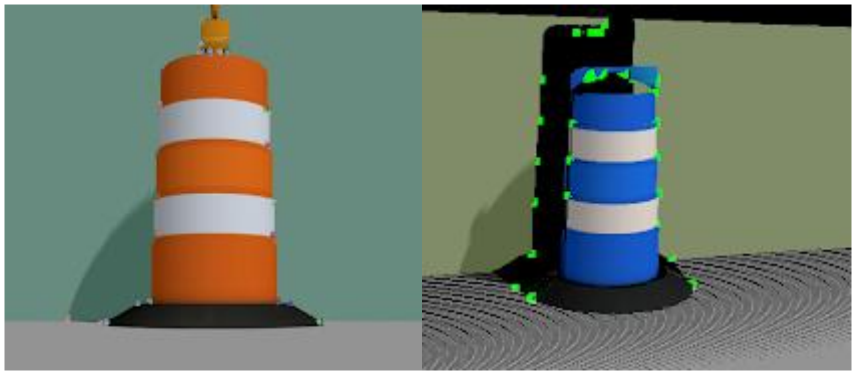


Figure 4.2 (a) – Results of ORB (left) and AGAST (right) key point extraction algorithms from traffic cone scene.

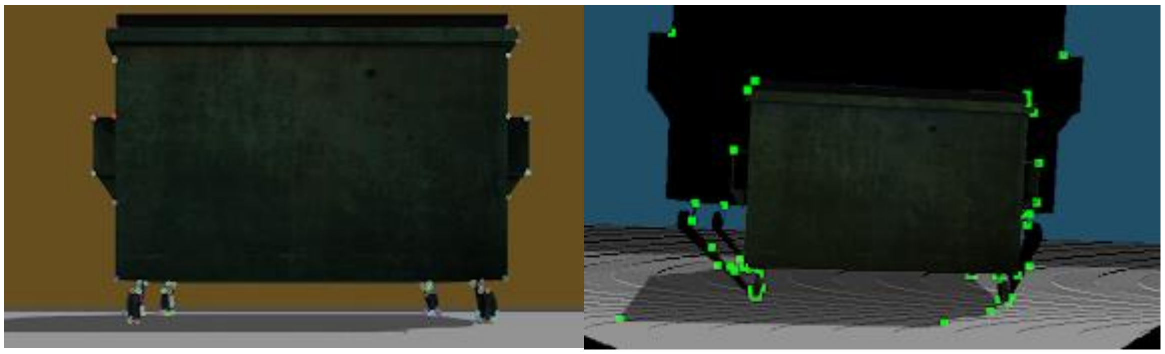


Figure 4.2 (b) – Results of ORB (left) and AGAST (right) key point extraction algorithms from garbage box scene.

4.2 Point cloud Descriptors for Extracted Landmarks (key points) and Point Cloud Registration

After finalizing on the key point extraction algorithm next step was to select a suitable key point descriptor to identify key points distinctly. The next step of point cloud registration will also depend on the descriptiveness of the selecting descriptor when performing tasks like correspondence matching in feature based and Iterative Closest Point (ICP) registration methods. Therefore, it was decided to test the quality of the descriptor by examining how well it performs in the point cloud registration task. As the set of point cloud feature descriptors, the available point cloud descriptors in the PCL library was used. Selected descriptors are,

- PFH (Point Feature Histogram)
- FPFH (Fast Point Feature Histogram)
- NARF (Normal Aligned Radial Feature)
- SHOT (Signature of Histogram of Orientation)
- RIFT (Rotation-Invariant Feature Transform)
- 3DSC (3D Shape Context)
- USC (Unique Shape Context)

The same experimental environment used for finalizing the key point extraction algorithm was used in this experiment as well. The specifications of the computer used in this analyzing is as follows,

- Processor – Intel Core i7
- RAM – 4GB
- Point cloud resolution – 480 x 640

The procedure used for point cloud registration is a method based on both feature based and a ICP based hybrid model was used as stated in the tutorial by D. Holz et al [6] on registration of point clouds captured by a Kinect sensor. According to the tutorial ICP performs well when two point clouds are merely aligned, so in this approach feature based registration was used to initially aligned the two point clouds to a certain extent and then feed the partially aligned point clouds to ICP algorithm to get a finer alignment. Moreover, applying a filter to smoothen out any noise from the input image was also suggested in the tutorial therefore applied the bilateral filter which smoothen amount areas with same color making edges more prominent as the preprocessing stage. The final procedure used for the point cloud registration is shown in the Figure 4.3.

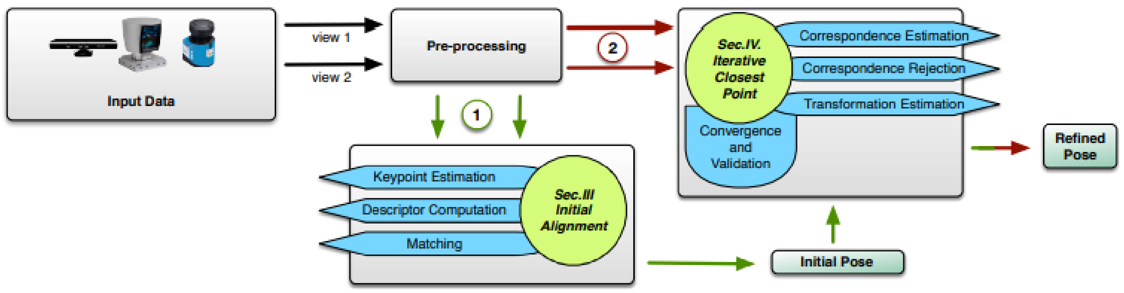


Figure 4.3 – Procedure of registering two point clouds [6]

Following the above procedure, the selected feature extraction algorithms were analyzed except the PFH descriptor due to it has a complexity of $O(n^2)$ which is not suitable for real time applications. In this experiment only rotation around its z axis was considered in this experiment (robots angular speed = 0.01 $rads^{-1}$) and used 150 key points per frame when mapping. Results are as follows,

Descriptor	**Descriptor size**	**Time taken**	**Registration Quality**
FPFH	33	1.9 seconds	Issues with similar scenes
NARF	36	1.5 seconds	Not satisfactory
SHOT	352	9.3 seconds	Issues with similar scenes
3DSC	1980	4.8 seconds	Issues with similar scenes
USC	1960	4.6 seconds	Issues with similar scenes
RIFT	32	17 seconds >	-

As we can see from the results trying to register point clouds using key points leads to errors in the map building process at a certain stage as shown in the Figure 4.4. There are two major causes for this issue,

1. If the robot come across a plane area where there are no sufficient details in the environment to capture any key points e.g. a plain color wall, map building process fails due to lack of key points.
2. Second scenario is, if robot come across an area where there is a pattern in the surrounding the mapping process will consider it as the same scene and map it to the same frame even though they are two different frames.

To overcome these issues robots odometry data has to be used at a certain point. Therefore, instead of using a method like feature based or ICP which is computationally expensive and also have some issues in certain occasions, a map registration process that uses the robots odometry data is decided to use. Key points and descriptors will be used to calculate the error that is generated in the odometry of the robot.

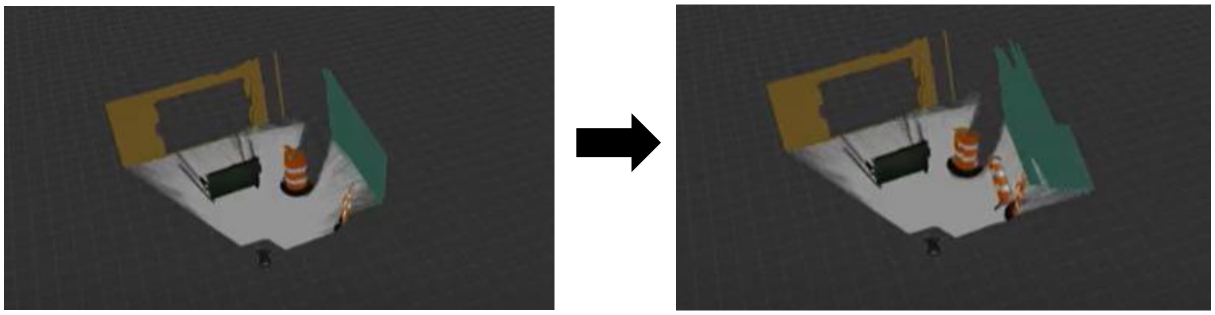


Figure 4.4 – Mapping failing when using ICP and Feature based approach

4.3 Point Cloud Registration using Robots Odometry Data

By default, ROS keeps track of the odometry of the robot and published through a topic called "/odom". As the first approach of registering point clouds using odometry data of the robot, data that is published through "/odom" topic was used to calculate the transform of the point cloud. There were few challenges in this approach,

- Processing and calculation of the key points of the point cloud take some time and taking the odometry data that is available after calculation of key points to calculate transform of the point cloud can be erroneous due to time delay.
- Robots coordinate axes is different from the coordinate axes of the point cloud that is captured by the sensor therefore, a coordinate conversion must be done when calculating the transform.

To address the first challenge a ROS node which will act as a database is created. Here, node will calculate transform for the each second and store the calculated transform until it is expired. A total of 180 values is stored in the database that is equal to transforms of three minutes. Data is stored as a form of a dictionary where key is the time that odometry message is received and the value id the transform that was calculated from that odometry data. This node also provides a ROS service where the point cloud registering node can request the transform by providing the time stamp of the captured point cloud. The overall architecture of the ROS nodes is shown in the Figure 4.5.

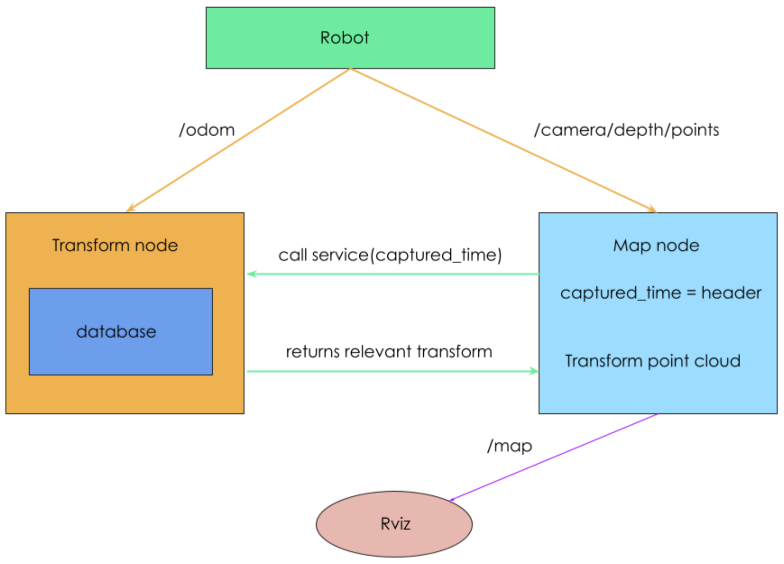


Figure 4.5 – Overall architecture of the ROS nodes to perform point cloud registration using robots odometry data.

As a solution for the second issue which was caused by having different coordinate axes between the point cloud frame and the robots frame, axes were switched accordingly when calculating the transform for the point cloud. Figure 4.6 shows the axes of the robot and the point cloud respectively.

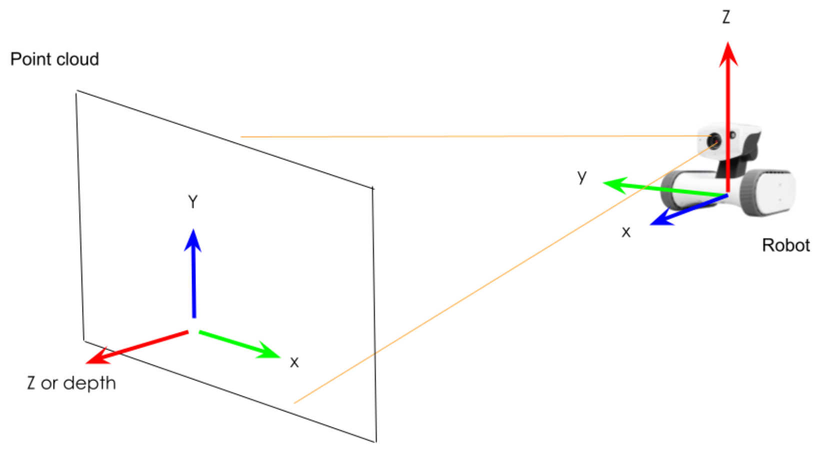


Figure 4.6 – Axes mismatch between point cloud and the robot.

The translation between the coordinate frames were done as follows,

- X axis of point cloud = - (Y axis of the robot)
- Y axis of the point cloud = Z axis of the robot
- Z axis of the point cloud = X axis of the robot

(Rotational angles were also changed respectively)

When calculating the transform for the final point cloud, the resultant transform was obtained by combining four matrixes of translation and rotation around three axes as given in the following equation,

$$Resultant\ Transform = Translation\ \times Rotation\ x\ \times Rotation\ y \times Rotation\ z$$

After the completion of the implementation point cloud registration or the map building process was tested in two new test environments. One environment represented a house hold environment while the other one represented a warehouse environment as shown in the Figure 4.7 (a) and Figure 4.7 (b) respectively.

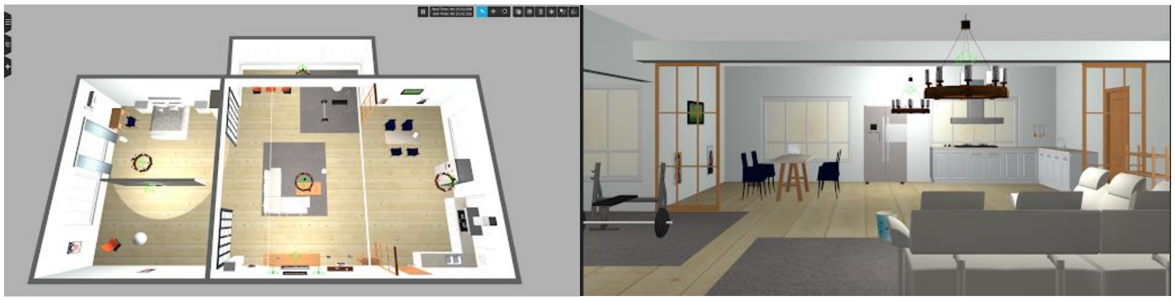


Figure 4.7 (a) – Testing environment representing a house

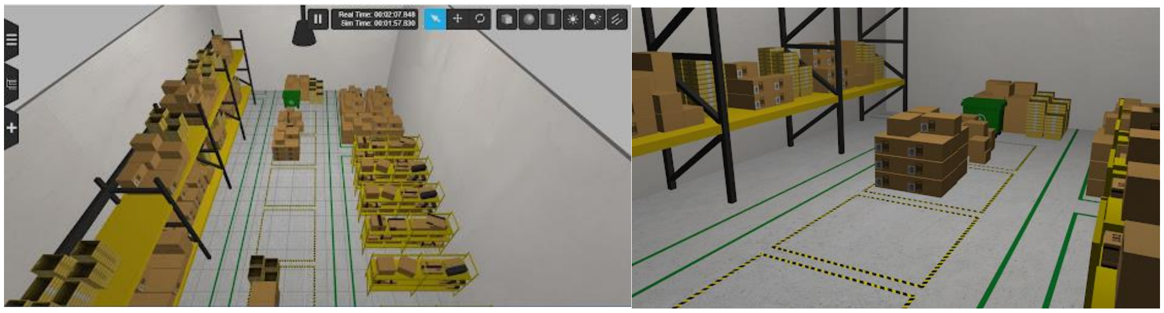


Figure 4.7 (b) – Testing environment representing a warehouse

When performing the mapping task issues that was occurred in the previous attempt with similar scenes did not occurred and the mapping happened as expected. Some snapshots from the mapping process in two testing environments is shown in Figure 4.8.

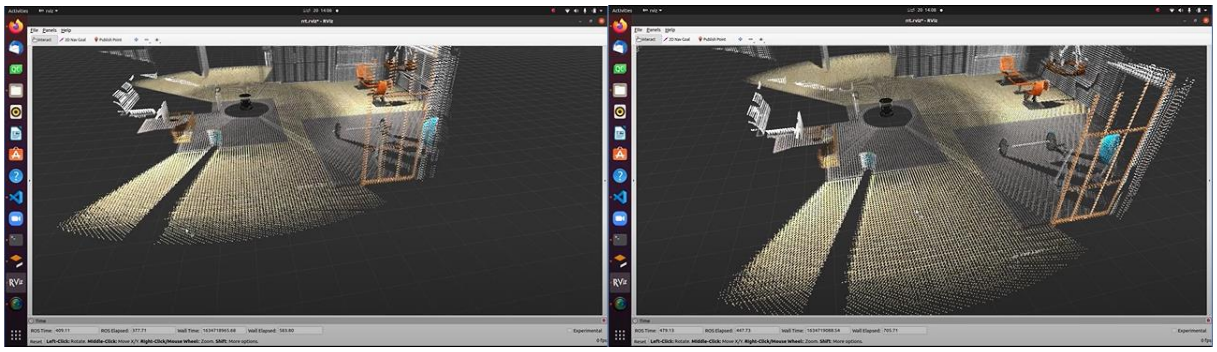


Figure 4.8 (a) – Snap shots of mapping of house environment –using odometry data

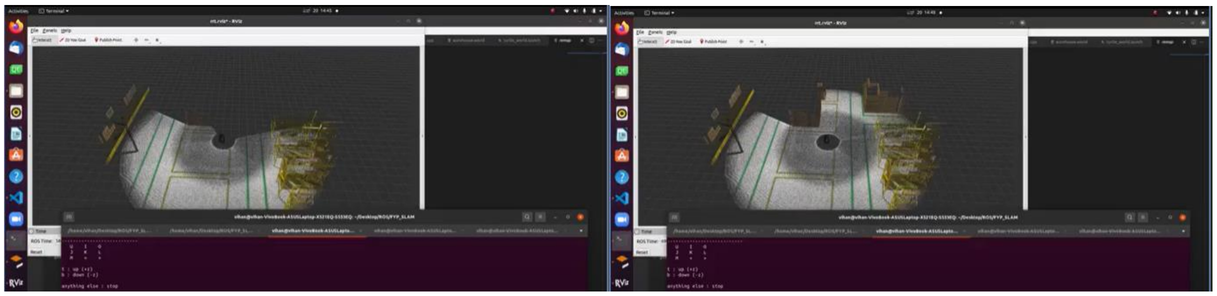


Figure 4.8 (b) – Snap shots of mapping of warehouse environment –using odometry data

Even though this approach performs well there is a minor drawback as well.

- Transformation from robots’ coordinates frame to point clouds’ coordinate frame should be hard coded.

After doing a research on how solve this issue, it was found that ROS provides a ROS node called “tf” which does a similar kind of task of calculating transforms among different coordinate frames in the robots’ world. Because transform is calculated by the ROS transforms are not required to hard coded into the code. So as the next step a point cloud registration process that is based on “tf” node of ROS was carried out.

4.4 Point Cloud Registration using ROS “tf” Node

In robotics separate frames are maintained for rigid bodies of the robot as shown in the Figure 4.9. Then by using individual transforms of the rigid frames, relative transformation between frames can be calculated using the chain rule.

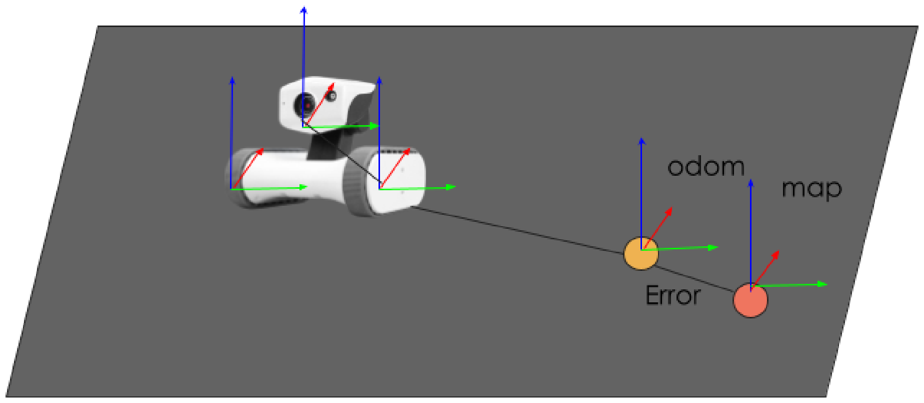


Figure 4.9 – Maintaining separate coordinate frames for individual rigid bodies in robotics

In the implementation an updated version of TF node was used which is known as “tf2”. This node provides several ROS services which can be used in calculating the transform between desired frames,

- Tf_listner: Listens to the transforms published by different frames and calculates the transform from source frame to the target frame that is specified by the user.
- Buffer: Buffers the calculated transforms for some time period.
- Tf_broadcast: Sometimes user needs to calculate a transform and publish it so that other frames can use that data in their calculations. Tf_broadcast can be used for this purpose.

When calculating transform for the point cloud in this case, source frame was obtained by the frame that is specified in the point cloud header sections and the “odom” frame which is the position where the robot was initialized was used as the target frame. To avoid any difficulties caused by not having transforms of the same time stamp a buffer of size 30 seconds was used.

By using the ROS transform calculation node similar that was achieved in the previous attempt was achieved as shown in Figure 4.10 also with a higher mapping rate. The issue of having to hard code the transforms manually was also resolved by using this approach.

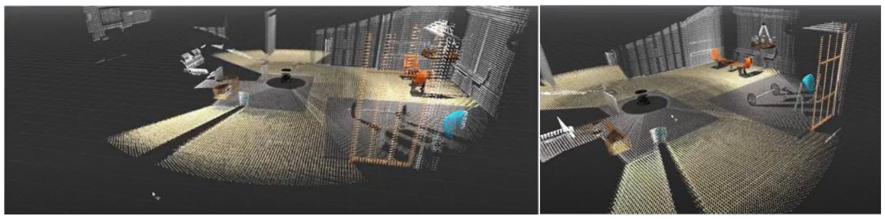


Figure 4.10 (a) – Snap shots of mapping of house environment – Using “tf” node

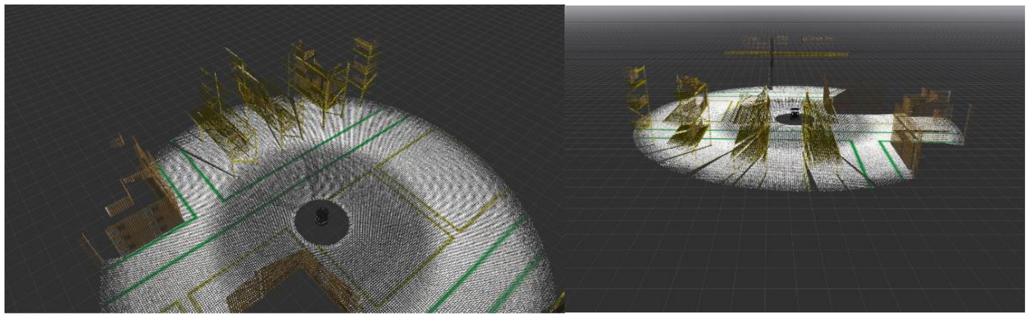
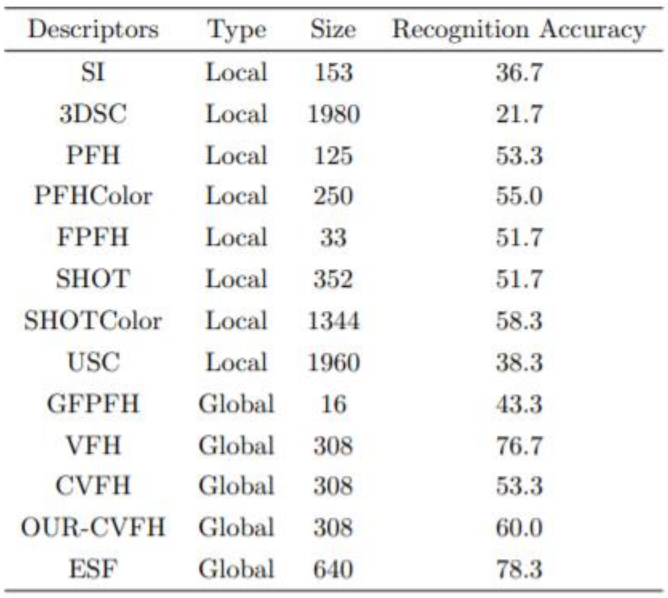


Figure 4.10 (b) – Snap shots of mapping of warehouse environment – Using “tf” node

4.5 Construction of Deformation Graph and Visualization

After successfully completing the point cloud registration phase as the next task, calculating any accumulated odometry error and deforming the map according to the error which is the final phase of the project was started. The first step of this task is to construct the deformation graph which is like a skeleton of the map that can be used to calculate error and calculate the deformation that needs to be carried out to minimize that error.

For the construction of the deformation graph key points that is extracted from the frames is used (10 key points from one frame). Feature descriptors of the key points also important because uniquely identifying each key point and the size of the descriptor will also determine the space required for the graph as well. From the results experiment did in the first attempt of the point cloud registration using feature based or ICP based approach and by considering the work done by X. F. Han at el. On analysis of point cloud descriptors which shown in Figure 4.11, FPFH feature descriptor was decided to use.



Descriptors	Type	Size	Recognition Accuracy
SI	Local	153	36.7
3DSC	Local	1980	21.7
PFH	Local	125	53.3
PFHColor	Local	250	55.0
FPFH	Local	33	51.7
SHOT	Local	352	51.7
SHOTColor	Local	1344	58.3
USC	Local	1960	38.3
GFPFH	Global	16	43.3
VFH	Global	308	76.7
CVFH	Global	308	53.3
OUR-CVFH	Global	308	60.0
ESF	Global	640	78.3

Figure 4.11 – Dimensionality and recognition accuracy of descriptors [5]

When building the deformation graph a dictionary data structure was used. Used the descriptor as the key and the coordinates where that descriptor was observed used as values. when a similar descriptor is encountered it will check with other coordinates where the descriptor was observed and if a coordinate closer to the current coordinate is encountered it will calculate the difference between those two points and broadcast as the error and deform the map.

To visualize the building of the deformation, graph all the detected key point locations is published as a separate cloud under the topic "/graph". Additionally, two ROS nodes were created to perform following tasks,

1. Graph construction node – subscribe to the "/graph" topic and writes all the coordinates to a test file.
2. Graph visualization node – reads the text file in regular intervals and construct the graph with robots' movement dynamically.

The deformation graph that was build when mapping the house environment is shown in Figure 4.12.

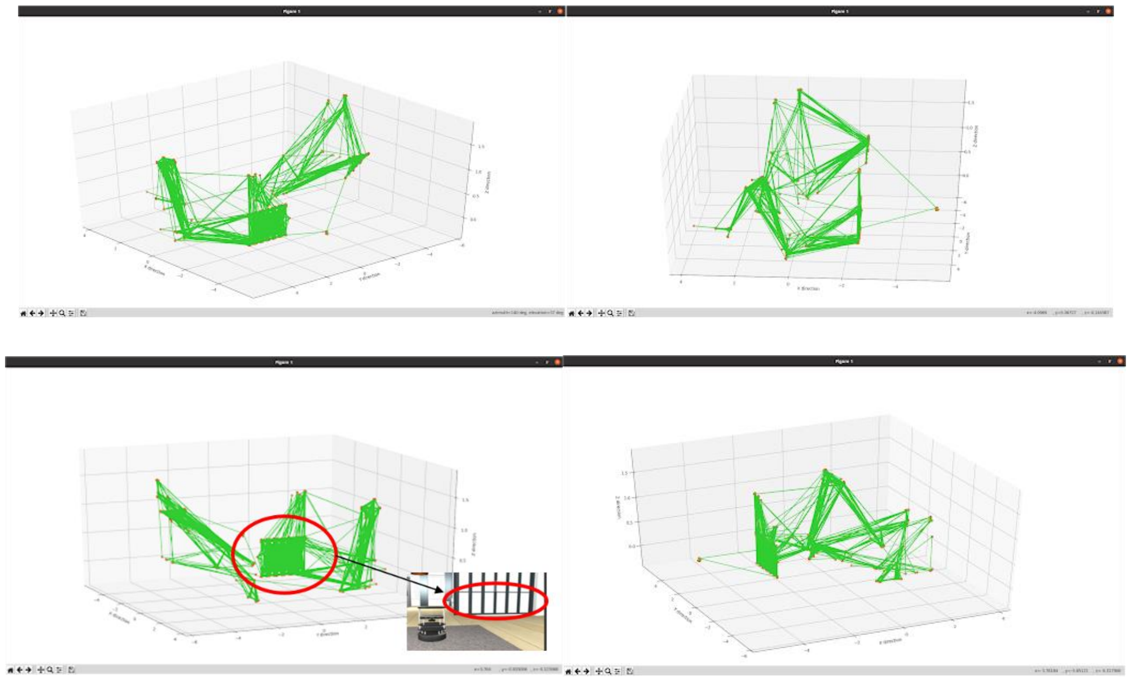
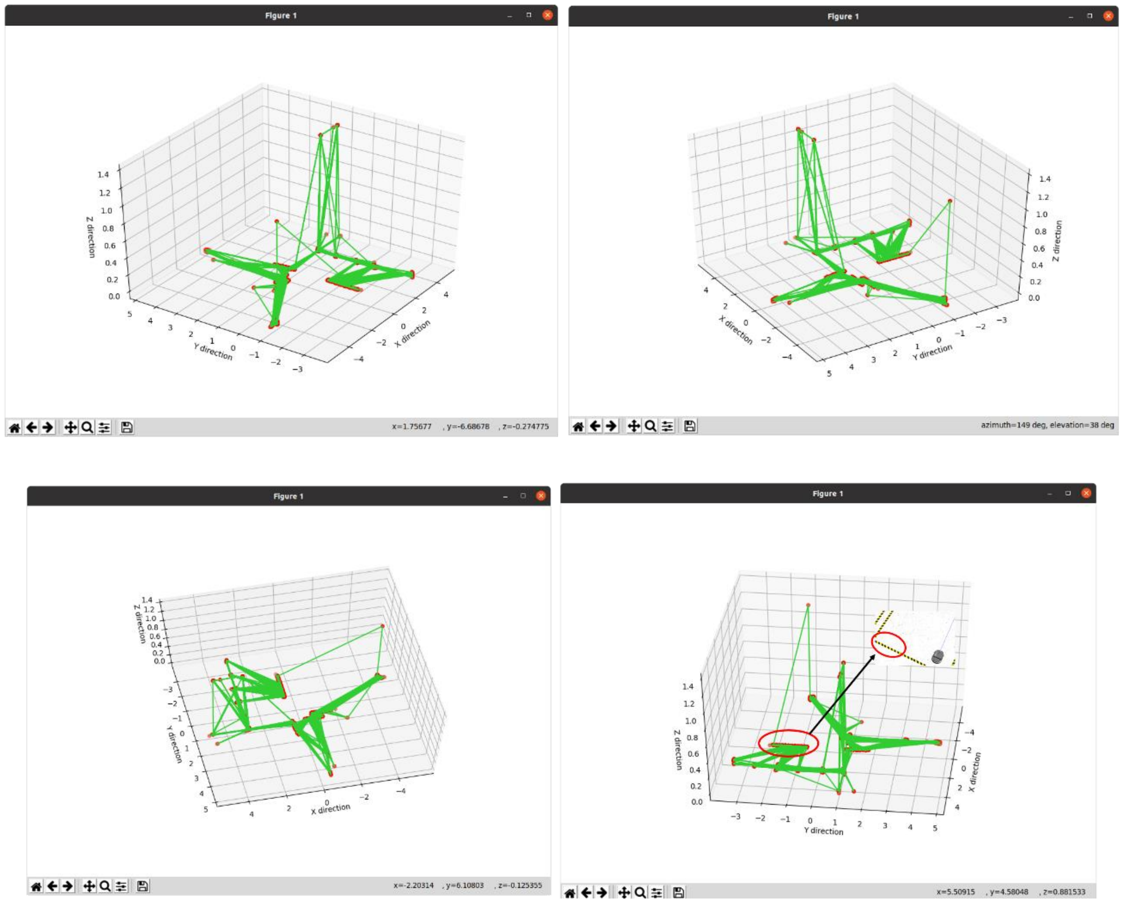


Figure 4.12 (a) – Deformation graph created when mapping the house environment (360^0 Rotation around its initial axis)



x=1.75677 , y=-6.68678 , z=-0.274775

azimuth=149 deg, elevation=38 deg

x=-2.20314 , y=6.10803 , z=-0.125355

x=5.50915 , y=4.58048 , z=0.881533

Figure 4.12 (b) – Deformation graph created when mapping the warehouse environment (360^0 Rotation around its initial axis)

4.6 Revised Methodology

Because of the alternative methods that was taken in order to achieve much more reliable system proposed methodology was modified as shown in the Figure 4.13.

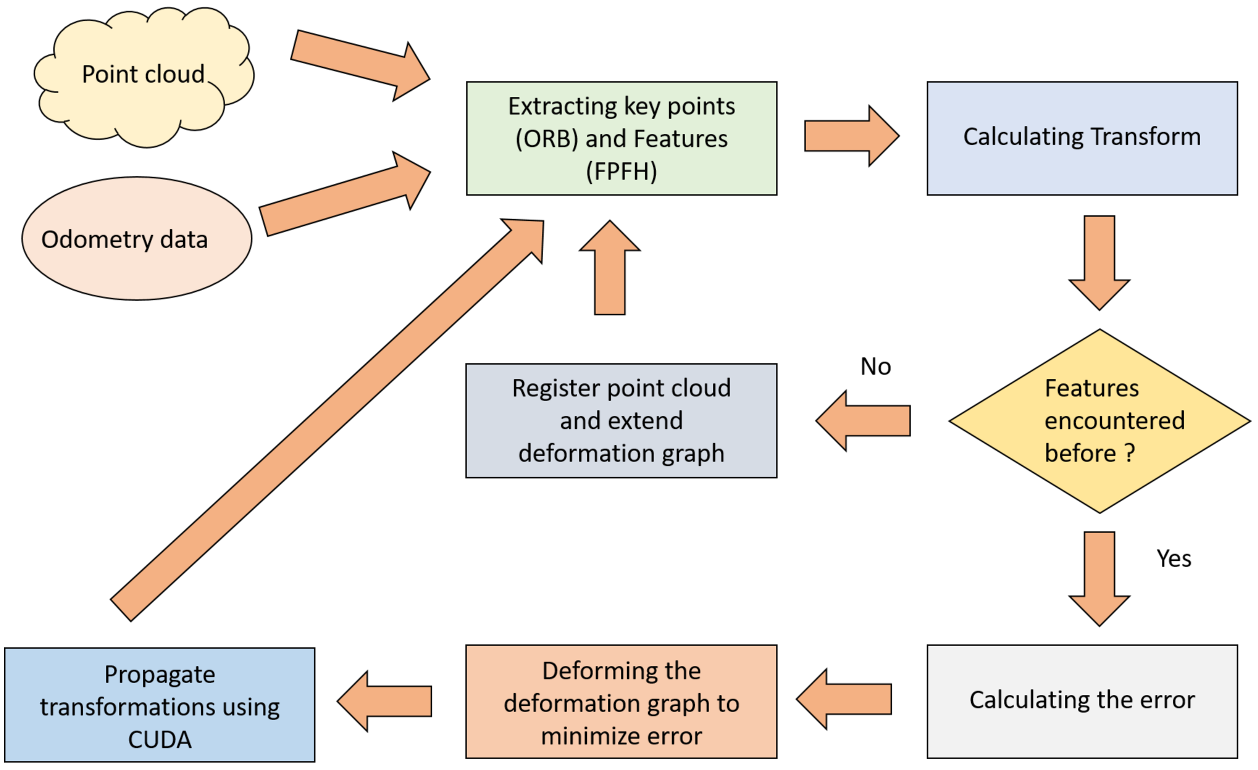

Point cloud
Odometry data
Extracting key points (ORB) and Features (FPFH)
Calculating Transform
Features encountered before ?
No
Register point cloud and extend deformation graph
Yes
Calculating the error
Deforming the deformation graph to minimize error
Propagate transformations using CUDA

Figure 4.13 – Modified methodology

5. Work to be Done

Up to now the front of the SLAM system where point cloud registration and map building process happens is completed and building deformation graph in the back end part is also completed. Calculating the odometry error if exists when the same feature is encountered in same location, deforming the deformation graph accordingly, and lastly propagating the transformations to other points which are not there in the deformation graph using CUDA is to be done in next phase of the project. The completed work according to the proposed time line is shown in the Figure 5.1 (Completed work is shaded in green color).

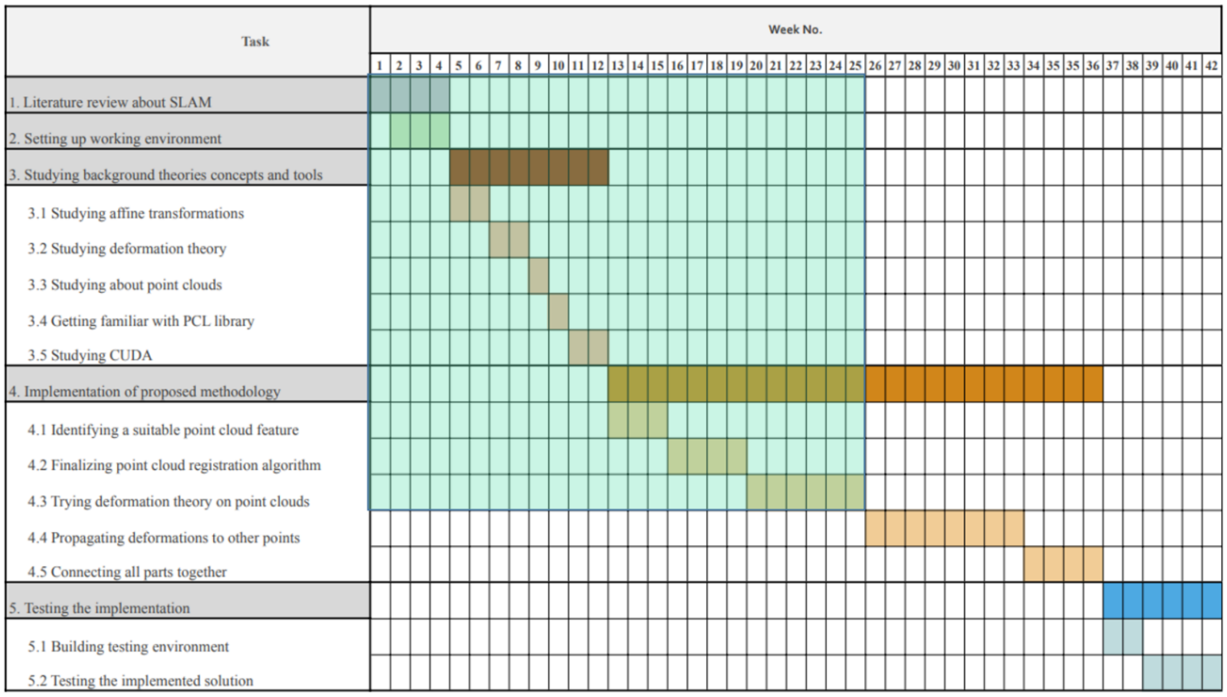


Figure 5.1 – Proposed timeline and completed work

6. References

[1]. C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016.

[2]. H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” IEEE Robotics Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.

[3]. F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-d mapping with an rgb-d camera,” IEEE Transactions on Robotics, vol. 30, pp. 177–187, 2014.

[4]. L. G´ongora Velandia, R. Hernandez Bele˜no, O. Avil´es S´anchez, and J. Rosario, “Mapping of indoor environments using point cloud library (pcl),” International Journal of Applied Engineering Research, vol. 11, pp. 5704–5713, 01 2016.

[5]. X.-F. Han, J. Jin, J. Xie, M.-J. Wang, and W. Jiang, “A comprehensive review of 3d point cloud descriptors,” 02 2018.

[6]. D. Holz, A. Ichim, F. Tombari, R. Rusu, and S. Behnke, “Registrationwith the point cloud library - a modular framework for aligning in 3-d,” *IEEE Robotics Automation Magazine,* vol. 22, pp. 110–124, 12 2015.

[7]. A. Huletski, D. Kartashov, and K. Krinkin, “Evaluation of the modern visual slam methods,” in 2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), 2015, pp. 19–25. 1

[8]. C. Kerl, J. Sturm, and D. Cremers, “Dense visual slam for rgb-d cameras,” 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2100–2106, 2013.

[9]. R. Mur-Artal and J. D. Tard´os, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[10]. S. Rahman, M. S. Abd Razak, A. Mushin, R. Hamzah, N. abu bakar, and Z. Abd Aziz, “Simulation of simultaneous localization and mapping using 3d point cloud data,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 16, p. 941, 11 2019.

[11]. S. Robert, S. Johannes, and P. Mark, “Embedded deformation for shape manipulation,” ACM Transactions on Graphics, vol. 26, p. 07, 07 2007.

[12]. S. Salti, F. Tombari and L. D. Stefano, "A Performance Evaluation of 3D Keypoint Detectors," *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, pp. 236-243, doi: 10.1109/3DIMPVT.2011.37.

[13]. T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. B. McDonald, “Real-time large-scale dense rgb-d slam with volumetric fusion,” The International Journal of Robotics Research, vol. 34, pp. 598 – 626, 2015.

[14]. T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. Davison, “Elasticfusion: Dense slam without a pose graph,” in Robotics: Science and Systems, 2015.