Indoor Navigation System using Range Imaging and V-SLAM

A project report submitted in partial fulfilment of the requirements for the degree of

Bachelor of Engineering

By

Yash Turkar (Roll No. 7977)

Yashom Dighe (Roll No. 7928)

Under the guidance of

Prof. Sunil Surve

Prof. B. K. Mohan (CSRE, IITB)

Dr. Yogesh Agarwadkar (InfiCorridor Solutions Pvt. Ltd.)

DEPARTMENT OF COMPUTER ENGINEERING

Fr. Conceicao Rodrigues College of Engineering, Bandra (W), Mumbai –400050

University of Mumbai

Year: 2019-2020

CERTIFICATE

This is to certify that the following students working on the project "**Indoor Navigation System using Range Imaging and V-SLAM**" have satisfactorily completed the requirements of the project in partial fulfilment of the course B.E in Computer Engineering of the University of Mumbai during academic year 2019-2020 under the guidance of Prof. Sunil Surve (Ph.D.).

Submitted By:

Yash Turkar (7977)

Yashom Dighe (7928)

Prof. Sunil Surve

Guide

Prof. B. S. Daga

Head of the Department

Principal

PROJECT REPORT APPROVAL FOR B.E

This is to certify that the project synopsis entitled **"Indoor Navigation System using Range Imaging and V-SLAM"** submitted by the following students is found to be satisfactory and the report has been approved as it satisfies the academic requirements in respect of Major Project - I work prescribed for the course.

Yash Turkar (7977)

Yashom Dighe (7928)

Internal Examiner	**External Examiner**
(Signature)	(Signature)
Name:	Name:
Date:	Date:

Seal of the Institute

DECLARATION OF THE STUDENT

We declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission.

We also declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources.

We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the student

Yash Turkar (7977)

Signature of the student

Yashom Dighe (7928)

Date: April 2020

ACKNOWLEDGEMENT

We would like to thank our institute Fr. Conceicao Rodrigues College of Engineering for access to the laboratory equipment and InfiCorridor Solutions Pvt. Ltd. for sponsoring the project. We are grateful to Prof. B. K. Mohan of Centre of Studies in Resources Engineering, IIT Bombay for his guidance in image processing techniques. Finally, we thank Mr. Christo Aluckal for helping us digitise all our figures and diagrams.

ABSTRACT

Unmanned aerial and ground vehicles (UAVs and UGVs) today excel at autonomous navigation due to their reliance on GNSS sensors. GNSS sensors are an unviable option for navigation in indoor environments due to lack of direct line-of-sight with satellites and lack of environmental awareness. This project proposes an integrated solution for localisation of the vehicle in an indoor environment, path planning and obstacle avoidance. The proposed system uses range imaging sensors for depth estimation and visual odometry (visual simultaneous localisation and mapping) for pose estimation. Simulations show that the proposed system can be used in a complex indoor environment with optimal lighting conditions and low to medium clutter of obstacles.

GLOSSARY

GNSS	Global Navigation Satellite System
GPS	Global Positioning System
VSLAM	Visual Simultaneous Localization and Mapping
ROS	Robot Operating System
WPS	Wi-Fi Positioning System
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
LIDAR	Light Detection and Ranging
GTG	Go to Goal
FSM	Finite State machine
CC	Counter Clockwise
CW	Clockwise
AO	Avoid Obstacle

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Unmanned Autonomous Vehicles or mobile robots, also known as UAVs (Unmanned Aerial Vehicles) and UGVs (Unmanned Ground Vehicles) are highly effective for a multitude of applications. They are extremely customizable and can be designed to carry various payloads such as cameras, LIDAR sensors and a variety of other sensors (optical, thermal, multispectral and hyper-spectral imaging sensors). This results in these robots having applications in various fields like surveying, agriculture, archaeology, emergency response and disaster management, mapping and reconstruction, civil and military reconnaissance to name a few. The ability to carry payloads and cover large distances makes them ideal for cargo delivery in remote places which are difficult to reach and time consuming to access by roads e.g. to deliver medicines and vaccines to doctors in remote locations.

Due to their reliance on Global Navigation Satellite System (GNSS) sensors (GPS, GLONASS, etc.) for localisation, these robots perform excellently in outdoor environments. GNSS satellites broadcast their location and GNSS sensors use these broadcasted signals to triangulate their own location w.r.t to the GNSS satellites. Hence, direct line-of-sight is required for GNSS to operate reliably. Satellite imaging also provides a perfectly accurate and detailed representation of the earth thus providing the data required (road maps and locations) for navigation which is in turn used by these robots to localise themselves in their environment/s.

Localisation in indoor environments is much more challenging because of high complexity of obstacles and clutter. Lack of line-of-sight prevents GNSS to provide reliable information for the vehicle to localise itself, GNSS also fails to provide any details of possible obstacles in the environment. Although previous approaches to develop a robust indoor navigation system such as ultrasonic beacons (SEKI et al. 1998), WPS (Wi-Fi Position System) (PARK 2014), Bluetooth based approaches (SATAN 2018), etc. exist, all of them require specialised hardware to be setup in the indoor environment and hence are not dynamic in nature and cannot be used in unexplored areas. Moreover, they need to communicate with a base station which makes them unusable in remote inaccessible areas, areas such as mines or caves.

Micro UAVs and UGVs have potential in indoor applications such as but not limited to survey operations in dangerous environments and structural analysis. To achieve localisation in indoor environments a reliable and robust navigation technique is required, which does not rely on GNSS or any other beacon based system (WPS, Ultrasonic beacons); a system that is self-sufficient, dynamic and robust enough to be deployed in real-life scenarios with minimal setup and no prior knowledge of the indoor environment.

The proposed system uses depth data from range imaging sensors and pose data from visual odometry sensors (V-SLAM sensor) and generates pointset which can be easily referenced later. The system considers possible obstacles in the environment and can operate in complete isolation, i.e. without communication with a base station. The system proposed is lightweight and simple, making it ideal to be run on-board the vehicle using minimal computing power. This makes the system completely independent and highly dynamic as no setup is required prior to operation.

2. LITERATURE REVIEW

2.1 Key Outcomes of Literature Review

There are several ways to localize and navigate an environment. GNSS receivers being the most popular method of localization. GNSS receivers use satellite communication to estimate their location. The navigation task performed by GNSS receiver in outdoor environments is well known. The searching in two domains, frequency and time, brings estimates in the Doppler frequency and code delay. In case of clear visibility there are several strong satellite signals present and the tasks of acquisition, tracking, and position computation are relatively easy.

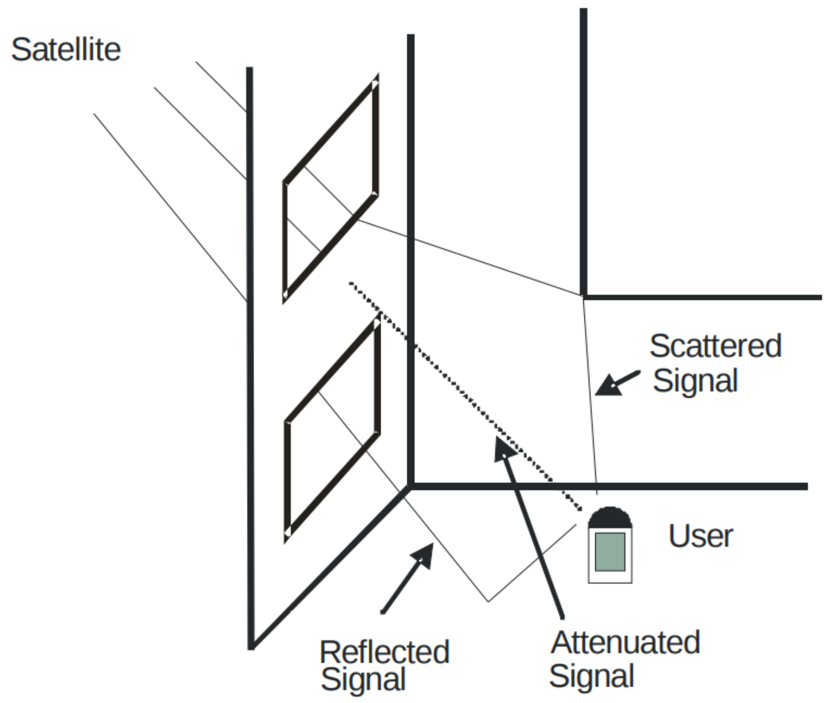

Figure 1 - GNSS in Indoor Environment (PURIČER, KOVÁŘ 2007)

The other situation is for the case of user receiver location in difficult environment, i.e. urban canyon or indoors. The signals coming from the satellites are obstructed by foliage, walls, and other structures (Fig. 1). The signal that can be used by the user receiver in such environment is mostly consisted of strongly attenuated direct signal and reflected (and/or scattered) signals coming usually by path of the least resistance. That is why two main phenomena, which indoor receiver must count with, are signal attenuation and multipath effect. (PURIČER, KOVÁŘ 2007)

Due to these challenges in using GNSS in indoor environments, other methods have been explored. Use of beacons has been widely adopted for indoor localization, beacons can be based on ultrasound (SEKI et al. 1998), Wi-Fi (PARK 2014) or Bluetooth (SATAN 2018) protocols. Beacon based localization systems are used in known environments, beacons use triangulation techniques to help the robot localize in the environment.

2.2 Research Gap

Most indoor navigation systems rely on beacons or external systems to localize a robot in an environment. Hence, these systems cannot be used in unknown or unexplored environments. These systems are also incapable of detecting and localizing obstacles in the environment as they can only calculate the position of an active system such as a robot communicating with the navigation beacons.

This generates a need to develop a system capable of localizing itself in an unknown and unexplored environment. Such a system should be capable of mapping, exploring and navigating an unknown indoor environment while avoiding obstacles.

2.3 Research Questions

This project aims to answer the following questions.

- What is the efficacy of developing a robust indoor navigation system using V-SLAM and range imaging?
- What is the relative efficiency that can be achieved by such a system?
- What are the target areas of application?
- What are the constraints and limitations of such a system?

2.4 Objectives and Problem Statement

2.4.1 Objectives:

- To assess the feasibility of developing a robust indoor navigation system using V-SLAM and range imaging
- To evaluate the efficiency of the proposed system
- To discover optimum use environment based on performance
- To pinpoint the limitations of the proposed system

2.4.2 Problem Statement:

"Exploring, mapping and navigating a complex indoor environment while avoiding obstacles and optimizing path"

Localisation in indoor environment is a challenging task. Exploring, mapping and navigating an indoor environment while avoiding obstacles in real time can be complicated. To efficiently solve this problem and deal with complexity, the problem statement is broken down into sub-problems. These sub-problems are solved discretely, the solutions are put together after unit testing and the final method is optimized to run with minimal computing power. Figure 2 shows the breakdown of the problem statement into sub-problems.

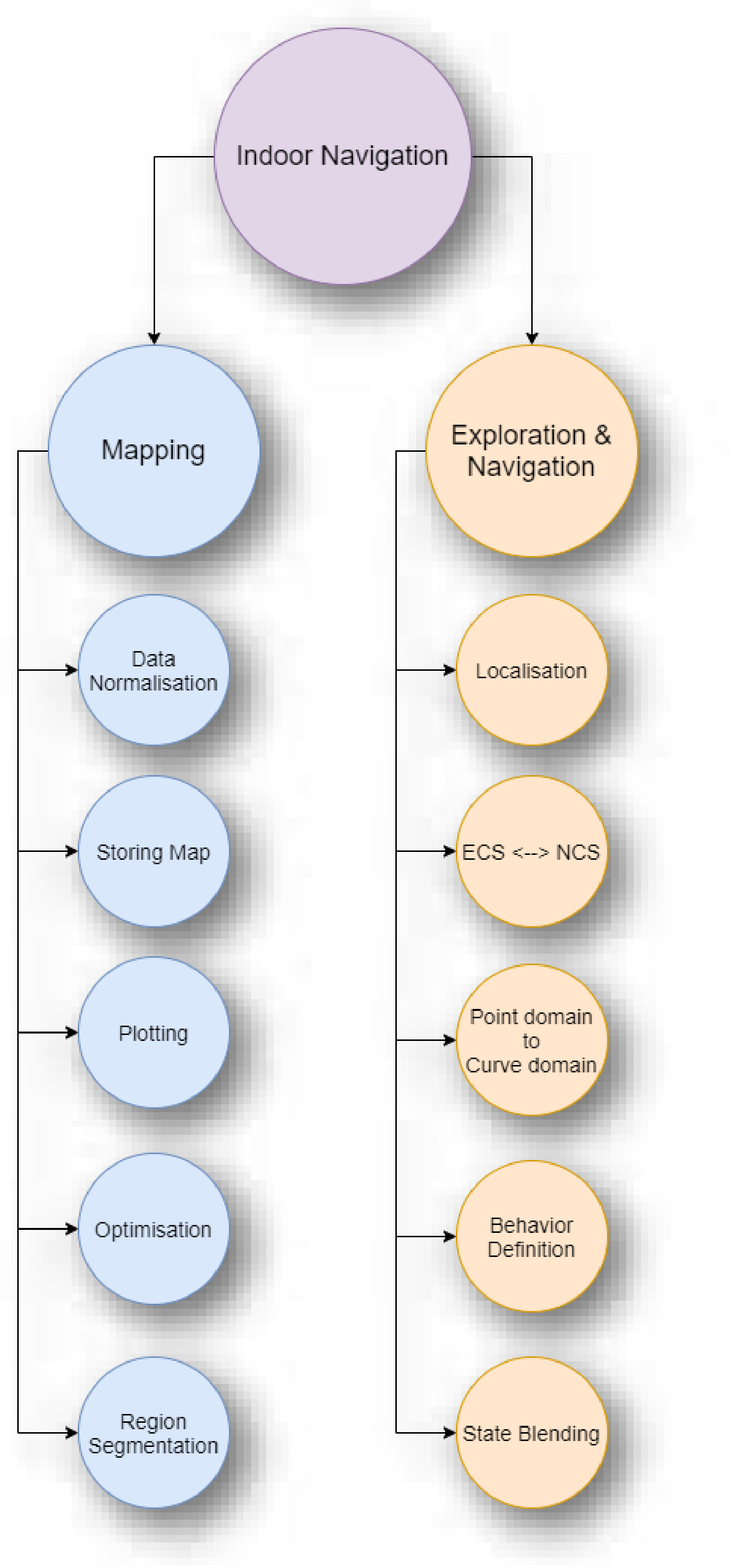

Figure 2 – Problem Breakdown

3. CONCEPTUALIZATION OF PROBLEM STATEMENT

3.1 Environment Mapping

a. *Data Normalisation* – Data acquired from the range imaging sensor and the pose estimation sensor needs to be normalised. The depth values per pixel provided by the range imaging sensor proposed are the normal distance from the sensor to the pixel. These need to be converted to radial values in door to generate the point-set (2D map).
b. *Storing Map* – The generated map or point-set needs to be stored in a custom data structure. This data structure needs to be lightweight, easy to manipulate and robust. This can be challenging as a concurrent approach is required.
c. *Plotting* - The generated map or point-set needs to be plot so as to show and verify the map in real time. This can be challenging as a concurrent approach is required.
d. *Optimisation* – Redundancies in point-set generation can result in excess data. Thresholding data and selective omission of areas is required for an optimised output.

3.2 Exploration and Navigation

a. *Localisation* – The system needs to be localised in its environment in order to explore its surroundings. Localisation in terms of the exploratory coordinate system (ECS) and w.r.t to its surroundings is merged to provide the system with complete sense of its location and surroundings.
   Traditionally localisation is achieved in an outdoor environment by using GNSS sensors. GNSS sensors are limited to outdoor use as there are several problems that make them unreliable for indoor operations.
b. *ECS↔NCS* – The exploratory coordinate system (ECS) is the coordinate system used when the system is exploring the environment. All the obstacles are stored using the ECS and distances are saved w.r.t the ECS origin. When a robot needs to navigate an explored environment, it may start from a different point in space, taking a new origin and a new coordinate system, the navigational coordinate system (NCS). In order to use the stored map (generated while exploration), ECS origin needs to be mapped with NCS origin. Once that is achieved, all obstacles can be avoided by the navigation algorithm.
c. *Region Segmentation* – Regions in an environment need to be segmented, i.e. in one map, different rooms and areas need to be identified as different mathematical constructs. Once these regions are segmented, path planning algorithm can be applied, and waypoints can be generated.

d. *Waypoint Generation* – Waypoints need to be generated in the workspace of the robot such that they do not coincide with any known obstacles and are weighted so as to make sure that the most optimum path is chosen every time.

e. *Point Domain to Curve Domain* - The 2D map that is generated while exploring is a point-set i.e. it is a list of all points that define the boundaries of obstacles or walls w.r.t to the origin in the ECS. Referring to this point-set while generating a path to navigate is computationally and mathematically infeasible and inefficient. This sequence of points needs to be converted to a mathematical equation of a curve that contains most of those points. This has two major difficulties.

- There needs to be a method to determine if there are different sets of points. This is a problem as while building the point-set there is no distinction made between different objects. E.g. If a curve fitting method is applied on a point-set of two walls that intersect at some angle, the resulting curve is an arc that is defined by those points. Figure 3 illustrates this example. For an ideal output, the two walls need to be treated as separate point-sets and curve fitting needs to be applied individually

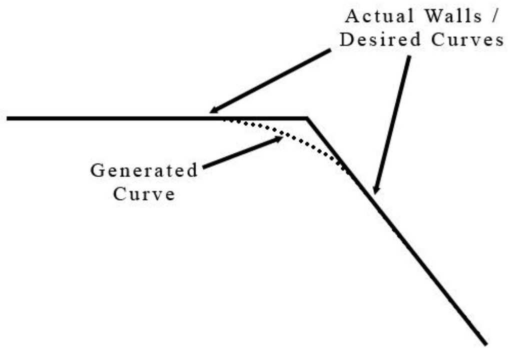

Figure 3 – Non-ideal vs ideal curve fitting

- Loss of detail - Curve fitting methods are approximations; they do not generate curves that pass through all the points. While this can be beneficial for dealing with noise in the sensor, it can eliminate details that are provided by outliers. E.g. If there is a minor (1-2 inch) irregularity in the wall (due to a decoration like a photo frame), a curve fitting method will simply eliminate this irregularity and the resulting 2D map will have a straight line. This loss of information could lead to a collision.

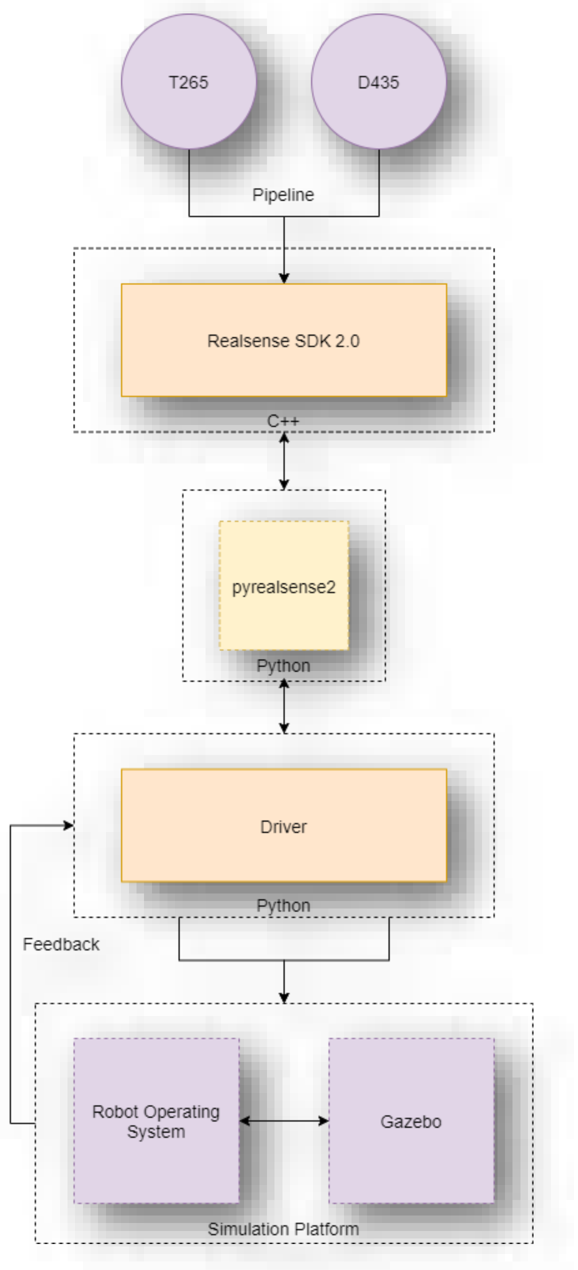

Figure 4 – Interdependencies

Figure 4 shows the interdependencies between the modules, APIs and platforms used for implementation of the proposed system.

The proposed system was implemented using Intel RealSense sensors, sensors such as but not limited to RealSense D435 and RealSense T265. To interface with the sensors, Intel's RealSense SDK was used along with a python wrapper. The algorithms used to explore, map and navigate are implemented using python. To verify the working of the system, a simulation platform consisting of ROS and Gazebo was used.

4. PROPOSED METHODOLOGY

4.1. Environment Mapping

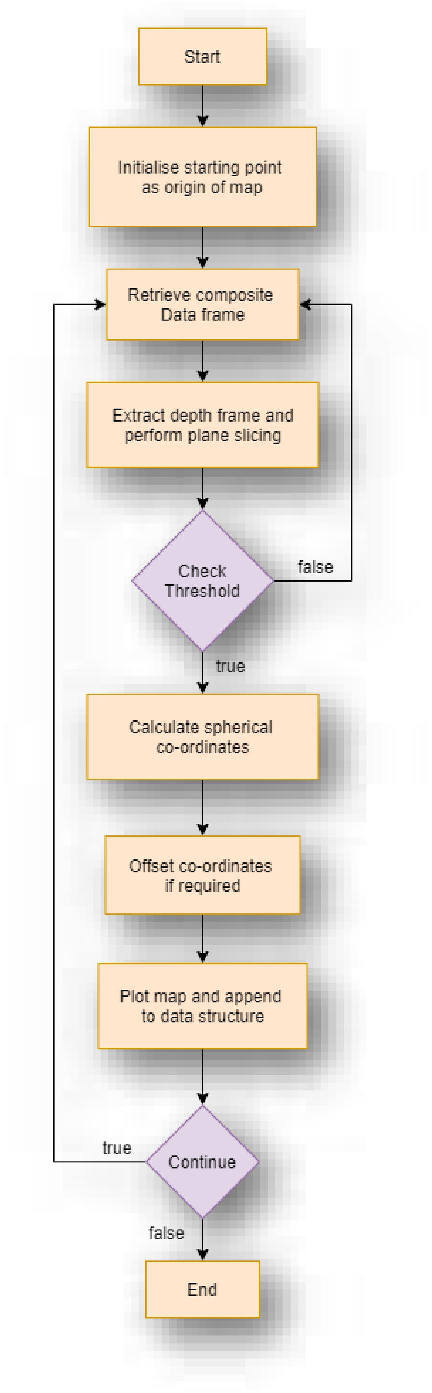
Start
Initialise starting point as origin of map
Retrieve composite Data frame
Extract depth frame and perform plane slicing
Check Threshold
false
true
Calculate spherical co-ordinates
Offset co-ordinates if required
Plot map and append to data structure
true
Continue
false
End

Figure 5 – Mapping Algorithm Flowchart

The range imaging sensor outputs data in a pipeline of frames. Each frame provides an R, G, B and depth value per pixel. Only one row in each depth frame is considered. Isolating and extracting only one row results in construction of a 2D-map at that altitude. This isolation is performed because the robot operates at a fixed altitude, making obstacles above and below the chosen altitude irrelevant. This is referred to as *plane slicing* and is demonstrated in Figure 6. The choice of row depends on the application and can be varied.

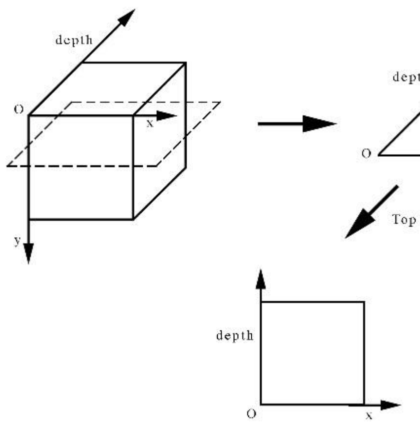

Figure 6 - Plane slicing

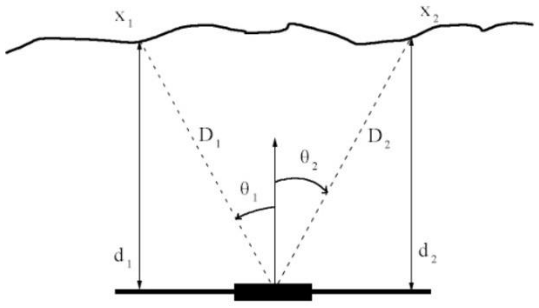

Figure 7 - Conversion to spherical coordinate system

4.1.1. Pre-Processing

The range imaging sensor provides depth values normal to the plane of the sensor. These values need to be converted to radial distance from the center of the sensor in order to convert spatial data to a spherical coordinate system for generating a visual representation. This is achieved. using basic Euclidean geometry and trigonometry

The sensor provides the normal distance d (refer Figure 7). By trigonometry, the radial distance D is the normal distance d divided by the cosine of □□

$$D = d/\cos\theta \tag{1}$$

The sensor has a horizontal resolution of n pixels. All these pixels have an associated distance d_1, d_2, d_3.... d_n and angles □$_1$, □$_2$, □$_3$....□$_n$ made with the vertical axis, which are used to calculate their respective radial distances D_1, D_2, D_3.... Dn. Thus, for every frame in the pipeline, the radial distances are calculated for all pixels in the chosen row.

□, for every pixel is calculated using the horizontal field of view h_{fov} of the sensor. The h_{fov}, which is a known value, is the angle between extreme pixels. The vertical axis is assumed to be aligned with center pixel and bisects h_{fov}. Counter-clockwise angles from the vertical axis to the left extreme are positive whereas clockwise angles from the vertical axis to the right extreme are negative. This makes the angle of the leftmost pixel to be $\left(\frac{h_{fov}}{2}\right)$ and the rightmost pixel to be $-\left(\frac{h_{fov}}{2}\right)$ and that of the center pixel to be 0. Thus, by elementary mathematics the generalized formula of □k for some kth pixel is given by

$$\theta_k = \left(\frac{h_{fov}}{2}\right) * \left(\frac{n}{2} - k\right) / \frac{n}{2} \qquad (2)$$

where, n is the frame width in pixels.

4.1.2 Proposed Method

The initial location of the robot is taken as the origin for the

map and the pose estimation sensor (visual odometry sensor). The pose estimation sensor returns the current position of robot relative to this origin (initial location).

Once a frame has been retrieved, two operations are performed

1. Thresholding and Selective omission
2. Offsetting

Thresholding is the process of checking if the robot has translated or rotated in the 2D space from its previous point of mapping and if the data in the current frame has been already mapped. The motion of the robot is verified using the pose estimation sensor which is expected to have an inertial measurement unit (IMU) that reports its attitude and position in the 3D-space relative to the starting point. The existence of knowledge is verified by referring to the existing pointset.

If thresholding determines that the robot's position or attitude has changed by a significant margin and that the knowledge of pixels in current frame does not exist in the data structure, spherical coordinates, as explained above, are calculated for those pixels. Here the thresholding function also *selectively omits* the pixels that have already been mapped in the case of an overlap as shown in Figure 8. Pixels between x_2 and x_3 are not processed again even though they are in the field of view, as they were already considered and added to the map previously.

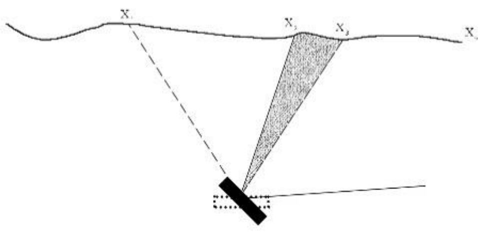

Figure 8 - Selective omission for rotation

After calculating the spherical coordinates of the pixels, they need to be *offset* because the range imaging sensor always gives depth from its current position. Therefore, if the robot translates or rotates, the radial distance or the angle with the vertical axis or both change relative to the starting point.

This needs to be compensated for and is done using the IMU and position data from the pose estimation sensor using standard homogenous translation and rotation matrices.

E.g. We create a translation matrix (Fig. 9)

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ posx & posy & 1 \end{matrix}$$

Figure 9 – Translation Matrix

After multiplication with current position matrix, we get the final coordinates of the points relative to the starting point.

Once the final coordinates are calculated, they are added as per the required format to the custom data structure and are plotted on the map using coordinate geometry for visual representation and verification.

4.2 Exploration and Navigation

Behavior based robotics is an approach of robotics that utilizes some basic behaviors programmed into the robots to react to their environment and solve problems within that environment. Instead of responding as per some precalculated model of the world they operate by reacting by combining or switching between behaviors or modes. The robots draw on their internal knowledge and basic feature set in order to resolve problems in the world that they operate in. These problems include “going to goal” or “avoiding obstacles”.

To navigate an indoor environment a robot needs 2 fundamental behaviours defined.

1. Go to Goal (GTG) – Drive the robot to the goal state
2. Avoid obstacles (AO) – Avoid slamming into things

The Control design task for both the behaviours is to pick a desired motion vector and set that vector equal to the input u

The controllers for these behaviours are obtained using control theory and solving the dynamics of the system.

Assuming that obstacles can be sensed and there is a way to specify goal state, GTG is implemented by minimizing the error between the current state and the goal state as shown in the Fig. 10. (State refers the position in this sense).

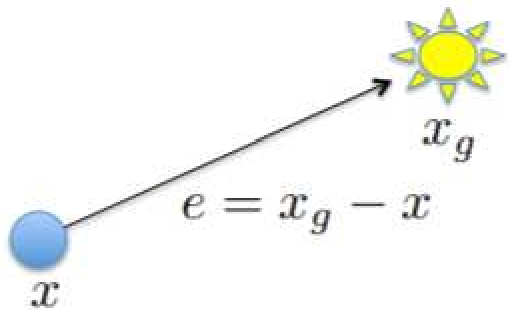

Figure 10 - e is the error between current state and goal state

In contrast AO is implemented by increasing or blowing up the error between the current state and the position of the obstacle as shown in the Fig. 11

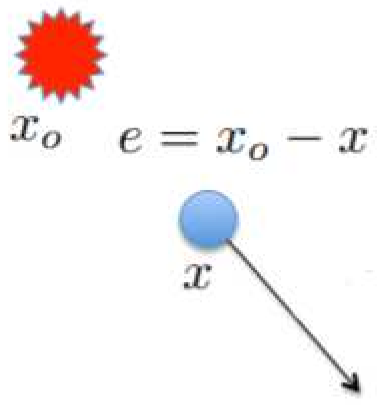

Figure 11 - e is the error between obstacle and current state

In order to combine these behaviours i.e. to reach a goal while avoiding obstacles a switching logic is needed to appropriately blend the behaviour and get the desired output.

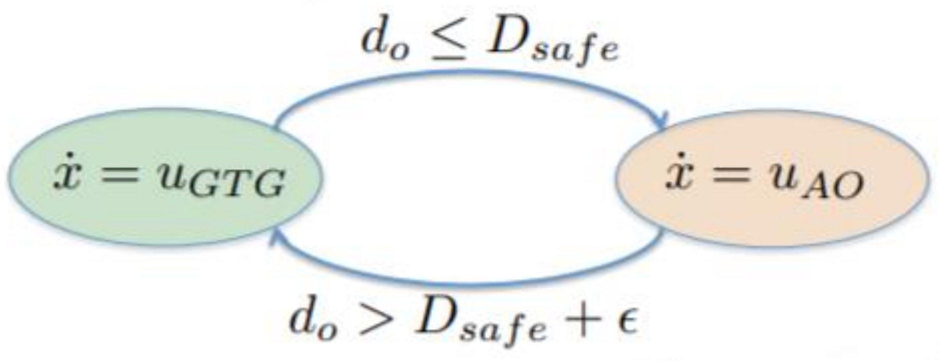

Figure 12 - FSM of Basic Controller

Let $\dot{x} = u_{GTG}$ and $\dot{x} = u_{AO}$ signify the inputs corresponding to the behaviours GTG and AO respectively. This means that these are two independent modes which we switch between.

Let $d_o$be the distance from the obstacle and D_{safe} be the threshold for maintaining the safe distance from an obstacle.

As implied by the state transition diagram (Fig. 12) we execute the GTG behaviour when we are sufficiently away from the obstacle and execute the AO behaviour when we are close to the object (close and far are defined by the D_{safe} parameter)

The $\in$ simply adds a tolerance to the distance as strict thresholds can lead to the Zeno's paradox issue due to infinite switching.

This model works when the obstacles are point objects but in the real world they are not. We define 5 classes of obstacles in increasing complexity.

1. Point objects – These objects have no size and are simply points in the world
2. Circular objects – These are same as point objects and the above model works for them perfectly
3. Convex objects – These obstacles are convex shapes (e.g. Rectangles)
4. Non-convex objects – Obstacles which are not convex in shape
5. Labyrinths – obstacles that are a combination of all the classes

The above-mentioned model works for class 1 and 2 but fails for the 3 classes. This is because the model makes them switch infinitely without any exit condition. This is demonstrated in the Fig. 13 below.

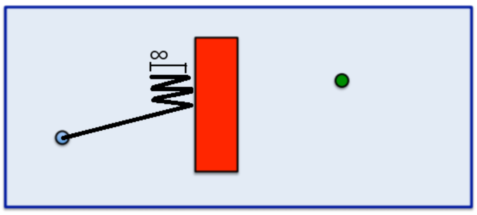

Figure 13 - Infinite switches without any exit conditions

The robot (shown by the blue dot) starts moving towards the goal (green dot). When it reaches near the red rectangle (convex obstacle), it switches to AO behaviour and moves in the direction away from the obstacle. When it is sufficiently far it executes GTG again and when it is near it switches back to AO. This repeats until the point of collision of the and the goal are colinear and results in the robot simply oscillating along that line. Something similar happens in case of non-convex objects and labyrinths.

In order to overcome this issue, we define an "Induced State" called follow wall that is to be executed that traces the boundary of an obstacle. Boundary refers to the boundary that is generated by the D_{safe} threshold. This implies that we trace the threshold distance.

Now to trace the threshold, we simply move in a direction perpendicular to the direction of AO behaviour as shown in Fig. 14

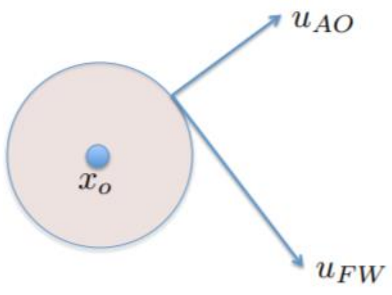

Figure 14 - Direction of FW is perpendicular to AO

Therefore, the input is,

$$u_{FW} = \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u_{AO} = \alpha R({}^{-\pi}/_{2}) u_{AO}$$

Where, R is the rotation matrix and α is a constant scalar. However, we are faced with another dilemma. The obstacle boundary can be followed in clockwise as well as counter clockwise direction meaning the angle of rotation can be $\frac{\pi}{2}$ or $\frac{-\pi}{2}$.

In order to decide which angle to choose from we take the help of the GTG vector. This can be done by taking an inner product of the FW and GTG vectors for C and CC directions and checking for the sign.

$$\langle u_{GTG}, u_{FW}^{c} \rangle > 0 \rightarrow u_{FW}^{c}$$

$$\langle u_{GTG}, u_{FW}^{cc} \rangle > 0 \rightarrow u_{FW}^{cc}$$

Where $\langle \quad \rangle$ is the inner product operator. The final issue in this problem is to decide when to stop following the wall and start moving to the goal i.e. recognising when do we have a clear path to the goal. The final problem is solved by maintaining calculating "progress" which simply means checking if the robot is closer to the goal now than when it started the FW behaviour and introducing the idea of clear shot which means can the robot clearly go to the goal directly. This idea of clear shot can be verified by checking the sign of inner product of the AO and GTG vectors because a positive inner product implies an acute angle.

Combining all the above ideas yield the following Finite State Machine (Fig. 15)

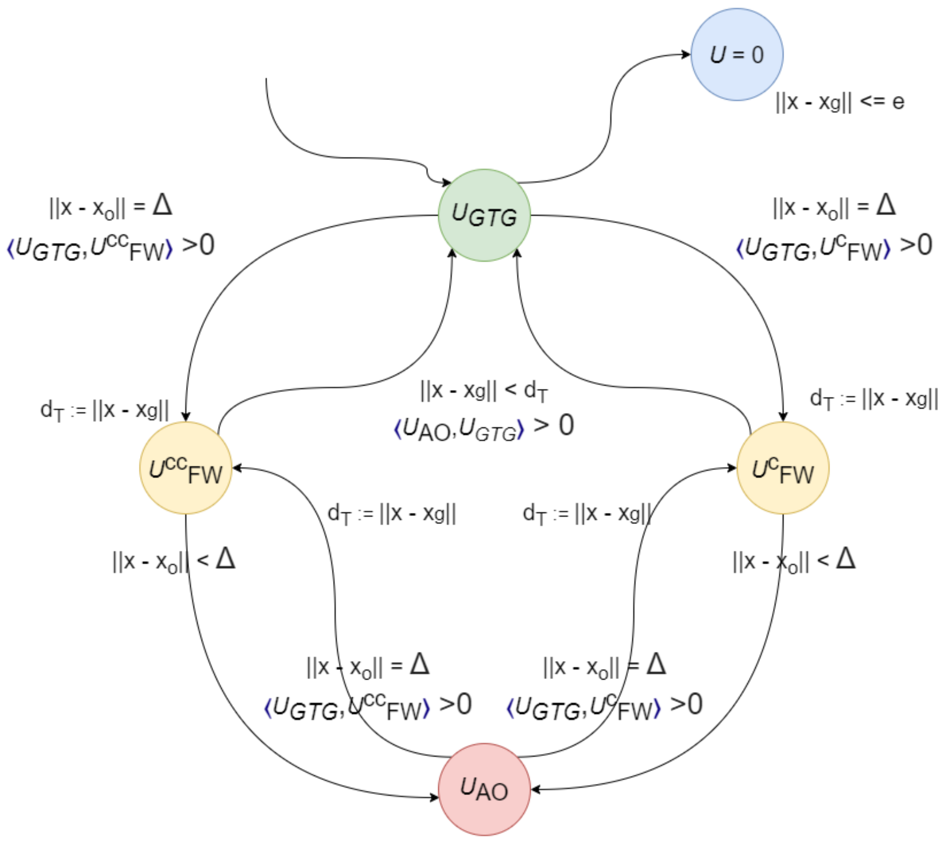

Figure 15 - State Transition Diagram for Behaviour Switching

Here, Δ denotes the threshold for safely avoiding the obstacles and $d_\tau := \|x - x_o\|$ is a reset used to recording position at the time for state transition for progress checking.

This is a complete model for the navigation controller. This FSM is utilised to explore and then negotiate the mapped environment as described in the follow sections.

4.2.1 Exploration

As defined, exploration is the process of navigating an unmapped world with the intention of building a map for future references and optimized navigation. This is achieved by generating arbitrary goal states near the current state of the robot using a semi-random generator. The goal state generator is semi-random because it discards points whose vicinity has been already covered in the map. The nearness is defined by the range of the depth sensor using for mapping. Semi arbitrary generation of goal states ensures that the entire world will be mapped given enough time.

4.2.2 Navigation

Navigation in a mapped environment utilises the above designed FSM to traverse a sequence of goal states. This sequence is generated using some heuristic to make sure that it satisfies the set condition for optimality.

4.2.3 ECS – NCS Mapping

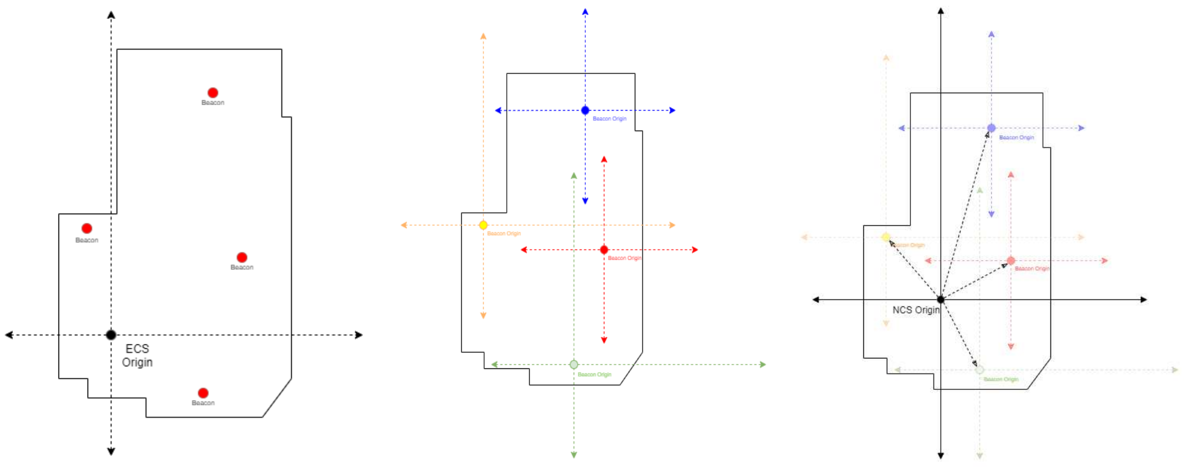

Figure 16 - ECS – NCS Mapping (a,b,c)

As listed under challenges, the first step of navigation is for the robot to localise itself in the 2D map once it starts i.e. to map this starting point(origin) to the origin of the ECS and shift the entire map from ECS to the NCS. In order to achieve this, visual beacons can be used. While exploring, certain visual beacons can be identified (objects like ceiling fans, switchboards, that are easy to detect and are unlikely to change place). As these beacons are identified in the ECS w.r.t. ECS origin (Fig. 16(a)), different copies of the map w.r.t. to different beacons as the origins are computed (Fig. 16(b)) using coordinate geometry and stored. During navigation, one/multiple of these beacons are in the environment and the distance to them is read from the range image sensor, the respective maps are loaded and then shift to be w.r.t. NCS origin (Fig. 16(c)). Once this is done, all the position readings from the pose estimation sensor will be absolute and the robot can know its position in the map.

5. IMPLEMENTATION

5.1 Environment Mapping

The method proposed in this project was implemented using Intel RealSense D435 depth camera as the range imaging sensor and Intel RealSense T265 tracking camera as the visual odometry sensor for pose estimation. The Intel RealSense SDK was used to interface with the sensors and the method was implemented using Python.

The camera has a horizontal field of view of 86 degrees and is set to a resolution of 640x480 pixels at 30 frames per second. This results in the value of n being equal to 640. Substituting these values in equation (2) makes the formula for θ_k as

$$\theta_k = 43 * (320 - k)/320 \tag{3}$$

5.1.1 Thresholding

- At initialization, the sensor's current position and angles are stored in a list which is then appended to a list called $prevList$.
- When the threshold function is called, the newer position and angle values are compared with the last appended value of $prevList$.
- If the differences in horizontal/vertical movement is greater than 3.5cm or angle moved is greater than 5 degrees, the threshold function returns true which results in the newer position and angle values to be appended to $prevList$ which will be used for further iterations.
- If criteria are not met, then the function returns False which doesn't update $prevList$.

5.1.2 Selective omission

$start\_p$: Start Pixel is an integer number signifying the pixel no. from which calculations should start from.

end_p: End Pixel signifies the pixel no. up to which calculations must take place.

Selective omission is implemented by limiting these values.

- This function takes two arguments, the g_angle (global Grand Angle) and the current yaw of the camera. The g_angle is the previous yaw value of the camera and is initialized to be 0.
- If g_angle is greater than yaw, it means that the camera was rotated in the CCW direction. Hence, the pixel values are calculated from the LHS. This makes $start\_p$ = 0, because it is the first pixel from the LHS. To calculate end_p, the difference in angle values is calculated. This difference is divided with the angle per pixel value which is $\left(\frac{h_{fov}}{frame\ width}\right)$= 86/640 = 0.134. Therefore, the end_p can be calculated as (g_angle – yaw)/0.134 and rounding it to the nearest integer number.

- If g_angle is lesser than yaw, it means that the camera was rotated in the CW direction. Hence, the pixel values must be calculated from the RHS. This makes end_p = 640, because it is the last pixel from the RHS. To calculate start, the difference in angle values is calculated. This difference is divided with the angle per pixel value which is $\left(\frac{h_{fov}}{frame\ width}\right)$ = 86/640 = 0.134. Therefore, an intermediate point $inter\_p$ can be calculated as $(g\_angle - yaw)$/0.134 and rounded to the nearest integer number. This value indicates the pixel difference. Subtracting this value from 640, gives the value of $start\_p$
- Cases are added where the difference between g_angle and yaw is 0 in which case the $start\_p$ is made 0 and end_p is made 640.
- Finally, the global g_angle is changed to be the current yaw. This value is used in subsequent iterations

5.2 Exploration and Navigation

The proposed behaviours and FSM were implemented in MATLAB on the sim-I-am simulator by GRITS Lab, Georgia tech. The simulator implements a differential drive robot that can be controlled as per the unicycle model i.e. by providing inputs for linear and angular velocities. The architecture of the simulator is as per Fig. 17

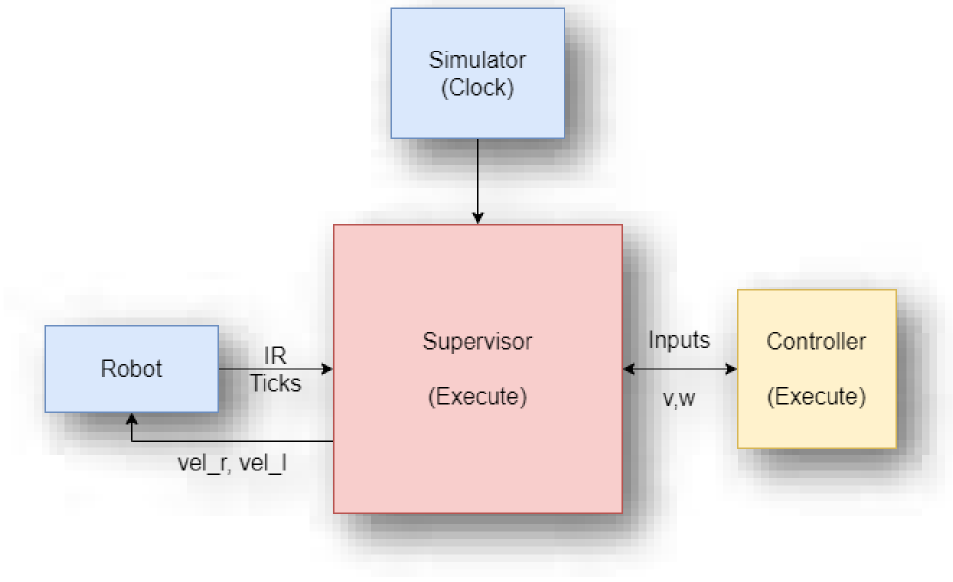

Figure 17 - Simulator architecture

The behaviours and the switching logic (i.e. the FSM) were programmed in the supervisor block. This simulator was use primarily used because it implements a similar IR based range finding as is describe in the mapping section.

5.2.1 Go to Goal Behaviour

The heading (angle), θ_g , to the goal location (x_g y_g) is calculated as follows.

1. Let u be the vector from the robot located at (x, y) to the goal located at (x_g y_g) then θ_g is the angle u makes with the x-axis (positive θ_g is in the counter clockwise direction). The vector u can be expressed in terms of its x-component, u_x, and its y-component, u_y. These two components and the $atan2$ function are used to compute the angle to the goal, θ_g

2. The error between θ_g and the current heading of the robot, θ, is calculated as follows. The error e k should represent the error between the heading to the goal θ_g and the current heading of the robot θ . $atan2$ is used to keep the error between [$-\pi$, π].

3. The proportional, integral, and derivative terms for the PID regulator that steers the robot to the goal are calculated .The robot will drive at a constant linear velocity v, but it is up to the PID regulator to steer the robot to the goal, i.e. compute the correct angular velocity.

5.2.2 Avoid Obstacles Behaviour

The IR sensors allow us to measure the distance to obstacles in the environment, but we need to compute the points in the world to which these distances correspond. Fig. 18 illustrates these points with a black cross

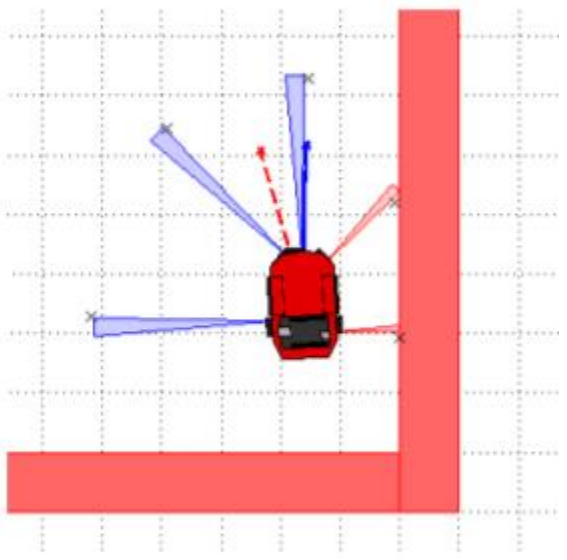

Figure 18 - IR range to point a transformation

1. The strategy for obstacle avoidance that we will use is as follows:
2. Transform the IR distances to points in the world.
3. Compute a vector to each point from the robot, u_1, u_2, . . ., u_9.
4. Weigh each vector according to their importance, $\alpha_1 u_1$, $\alpha_2 u_2$, . . ., $\alpha_9 u_9$.
5. Sum the weighted vectors to form a single vector, $u_{AO}= \alpha_1 u_1$ + . . . + $\alpha_9 u_9$.
6. Use this vector to compute a heading and steer the robot to this angle.

This strategy will steer the robot in a direction with the freest space (i.e., it is a direction away from obstacles). For this strategy to work, three crucial parts need to be implemented:

- Transform the IR distance measured by the sensor to a point in the reference frame of the robot.
- Transform the point in the robot's reference frame to the world's reference frame
- Use the set of transformed points to compute a vector that points away from the obstacle. The robot will steer in the direction of this vector and avoid the obstacle

5.2.3 Follow Walls Behaviour

1. First, a vector, $u_{fw,t}$, that estimates a section of the obstacle ("wall") next to the robot using the robot's right (or left) FOV is constructed as shown in Fig. 19

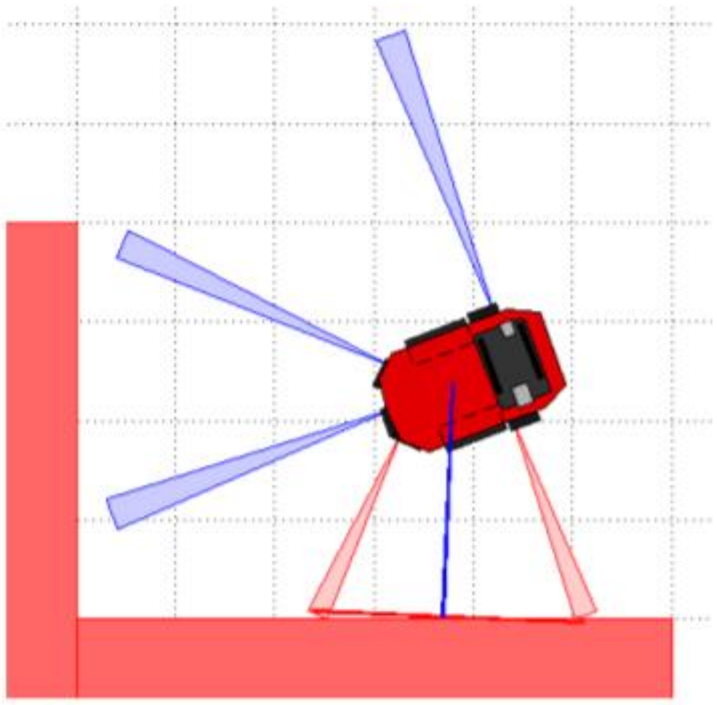

Figure 19 - Generation of **approximation** *of a wall*

2. A vector, $u_{fw,p}$, that points from the robot to the closest point on $u_{fw,t}$ is computed This vector is visualized as blue line in Fig. 19. and can be computed using a linear algebra.

3. The two vectors are combined, such that it can be used as a heading vector for a PID controller that will follow the wall to the right (or left) at some distance , d_{fw}.

4. A progress made event is implemented that determines whether the robot is making any progress towards the goal .This strategy is needed to realize that the robot is not making any forward progress and switch to follow wall to navigate out of the obstacle. This function returns true if
5. $\left\|\begin{bmatrix} x - x_g \\ y - y_g \end{bmatrix}\right\| < d_{progress} - \epsilon$

6. The sliding left and sliding right events are implemented, these serve as a criterion for whether the robot should continue to follow the wall (left or right) or switch back to the go-to-goal behaviour. While the lack of progress made will trigger the navigation system into a follow wall behaviour, we need to check whether the robot should stay in the wall following behaviour or switch back to go to goal. We can check whether we need to be in the sliding mode (wall following) by testing if $\sigma_1 > 0$ and $\sigma_2 > 0$, where

$$[u_{gtg} \quad u_{ao}] \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = u_{fw}$$

5.2.4 Finite State Machine

1. If **at_goal**, then switch to stop.
2. If unsafe, then switch to state **avoid_obstacles**.
3. If in state go_to_goal and at obstacle, then check whether the robot needs to slide left or slide right. If so **set_progress_point**, and switch to state **follow_wall** (with inputs equal to right or left depending on the results of the sliding test).
4. If in state **follow_wall**, check whether **progress_made** and the robot does not need to slide **slide_left** (or slide right depending on inputs). If so, switch to state **go_to_goal**, otherwise keep following wall

6. RESULTS AND DISCUSSION

The results generated by implementing the said method using the sensors mentioned earlier are discussed in this section and are compared with various versions of the method.

6.1 Environment Mapping

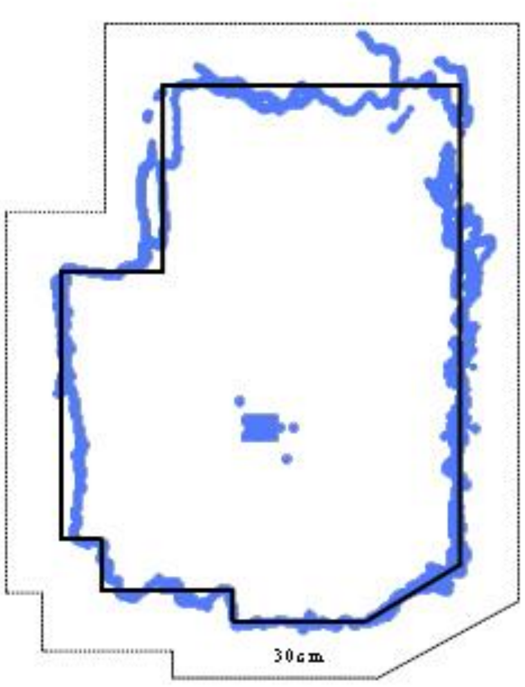

Figure 20 - Approximate blueprint and output stacked

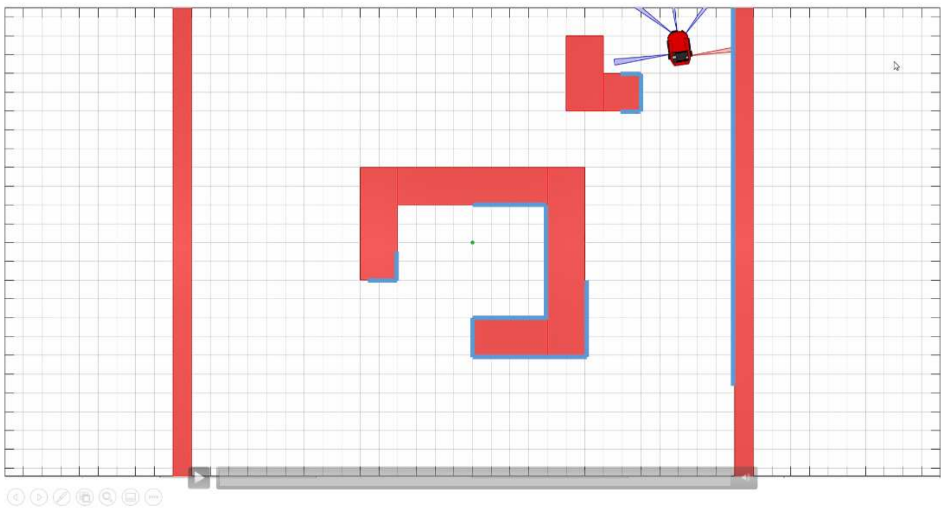

Figure 21 - Region Mapped by robot while exploring in a simulation (Marked in blue)

Fig. 20 demonstrates a map generated by the method proposed in this synopsis in blue. The black overlay represents a very approximate blueprint of the controlled environment, a room. The results show that the method performs well in areas with optimum lighting conditions. The dotted line outside the stacked boundary shows the extent of the overshoot, i.e. the points generated by the method that are further away from the origin than expected.

Although it seems like the generated output (Fig. 20 Blue Region) is not as perfect as the blueprint (Fig. 20 Black Region) itself, one must take into consideration the objects in the room. All the jagged edges and discontinuities observed are generated due to obstacles and objects present in the room. As the aim of the method is to generate a knowledge base, that is, a map that can be used to navigate the environment. It is not practical to ignore these discontinuities as that might lead to omission of certain obstacles in the environment.

Fig. 21 shows the region potentially mapped by a robot in a simulation after one iteration of moving from start to goal position. The information collected by multiple iterations will be accumulated to form a knowledge base which can be used in the future for navigating efficiently.

6.2 Exploration and Navigation

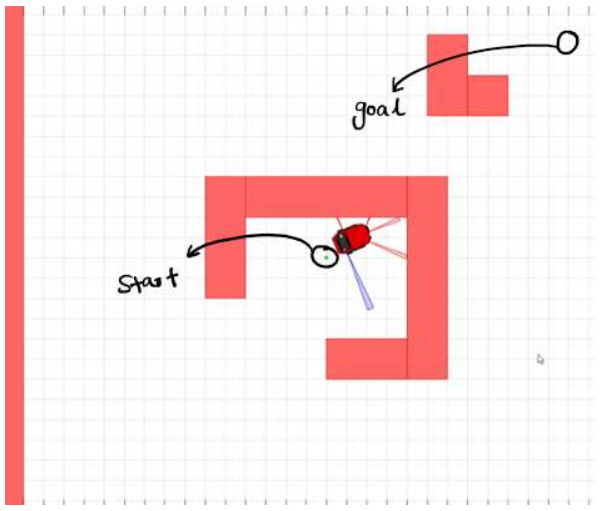

Figure 22 - Robot Starts from Initial Position

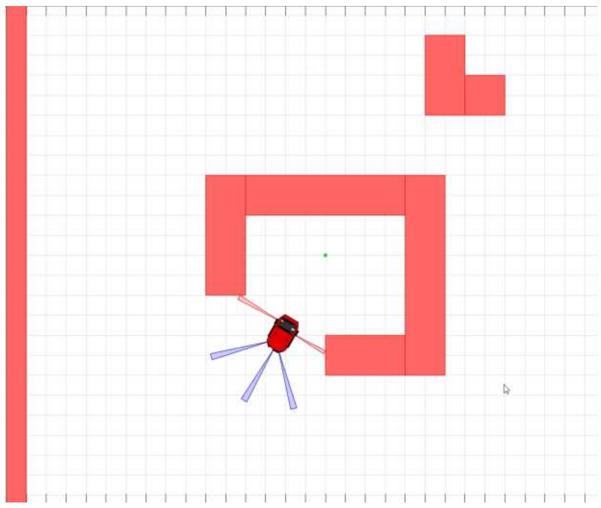

Figure 23 – Robot successfully exits the non convex obstacle

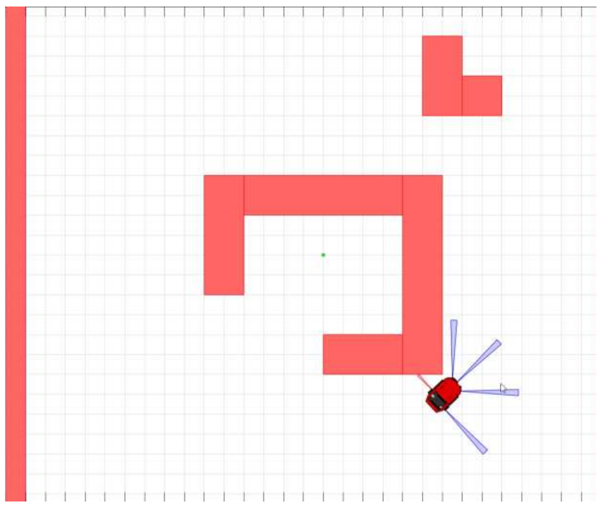

Figure 24 – Robot successfully navigates around the obstacle towards the goal

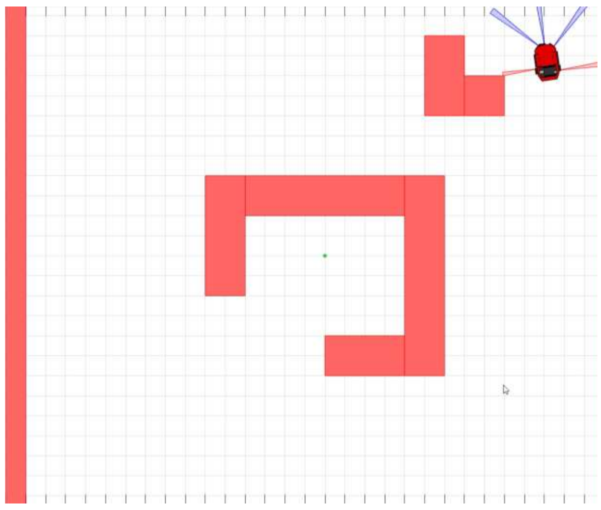

Figure 25 – Robot has reached the goal

The algorithm for navigating has been implement and tested in the most complex environment as defined in section 4.2 i.e. a labyrinth. The results generated in the simulated environment show that the robot can successfully negotiate the environment without slamming into any obstacles. This result is satisfactory for demonstrating that the algorithm works and can be implemented on a physical robot for further testing and validation in the real world.

7. CONSTRAINTS AND LIMITATIONS

- Lighting Conditions - The method performs well under moderate lighting conditions, extreme differences in exposure can result in discontinuities in the output generated by the method
- Vibrations – The method is not capable of handling vibrations that affect the data while it is captured by the sensor. A stable hardware platform is required in order to achieve optimal results.
- Distortions – The method relies on the sensor used to capture data for distortion correction. Pincushion distortions on the extreme ends of the depth frame were observed. Hence, distortion correction must be applied to the frame before being used.

8. CONCLUSION

This project proposes a dynamic and novel system for indoor navigation using range imaging and visual simultaneous localisation and mapping (V-SLAM). The system proposed is capable of autonomously exploring an indoor environment while mapping and tracking obstacles in real time. The system is also capable of navigating a previously explored environment by generating waypoints and optimising path.

This report discusses the efficacy of the system proposed and provides results generated by implementing the system in a controlled environment.

The results generated show that the proposed system can be used in real-life scenarios and is able to generate a fairly accurate map of the environment while exploring the environment in the test case while still having a scope for further development.

9. FUTURE SCOPE

- Exposure Correction – While imaging, if there is an area in the image which is overexposed or underexposed it will cause an error in the ranging data. Thus, a processing technique is required that eliminates this error by either normalizing the exposed area or ignoring the exposed region entirely.
- Selective Removal of Pixels – Manually or automatically pass range of values between which no mapping is generated. The benefit of this feature is the fact that, if there is a requirement to ignore certain ranges of pixels, then they must be eliminated despite meeting any prior mapping criteria.
- Extended Awareness - Within the generated pointset, we must be able to determine not only the existence of a potential obstacle and localize it within the environment but also classify and assign a weight to it based on the probability that the object is likely to move making its localization in the environment volatile. E.g. Small furniture like chairs have a non-trivial probability of changing their position in a room whereas storage cabinets are unlikely to move in the short run. This can be used to be aware of humans in the environment and safely operate alongside them without posing as a threat.
- Implementation of Region Segmentation – To optimize navigation, waypoints can be generated which act as sequential goal states. Segmented regions can provide a framework to generate waypoints.

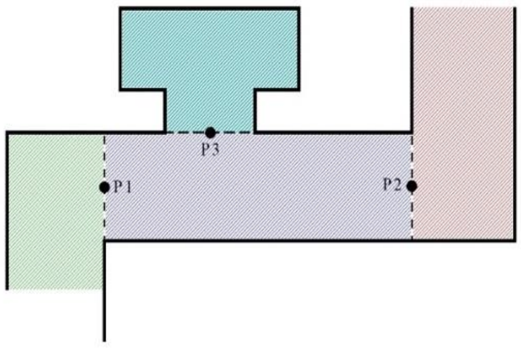

Figure 26 - Region Segmentation

REFERENCES

[1] J. Vandorpe, H. Van Brussel and H. Xu, "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder," Proceedings of IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 1996, pp. 901-908 vol.1.

[2] Morris, William & Dryanovski, Ivan & Xiao, Jizhong. 3D Indoor Mapping for Micro-UAVs Using Hybrid Range Finders and Multi-Volume Occupancy Grids (2010).

[3] Hiroaki Seki, Yasuo Tanaka, Masaharu Takano, Ken Sasaki, "Positioning System for Indoor Mobile Robot Using Active Ultrasonic Beacons", IFAC Proceedings Volumes, Volume 31, Issue 3, 1998, Pages 195-200.

[4] Y. Park, "Smartphone based hybrid localization method to improve an accuracy on indoor navigation," 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, 2014, pp. 705-708.

[5] A. Satan, "Bluetooth-based indoor navigation mobile system," 2018 19th International Carpathian Control Conference (ICCC), Szilvasvarad, 2018, pp.

[6] Xiadong Li and Nabil Aouf , “Cooperative vSlam Based on UAV application”, 2012 IEEE International Conference on on Robotics and Biometrics

[7] R. C. Luo, M. Hsiao and C. Xie, "Sensor fusion based vSLAM system for 3D environment grid map construction," 2013 IEEE International Symposium on Industrial Electronics, Taipei, 2013, pp. 1-6.

[8] J. Cheng, X. Zhu, W. Ding and G. Gao, "A robust real-time indoor navigation technique based on GPU-accelerated feature matching," 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcala de Henares, 2016, pp. 1-4.

[9] X. Li, N. Aouf and A. Nemra, "3D mapping based VSLAM for UAVs," 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, 2012, pp. 348-352

[10] A. M. Pinto, P. Costa, A. P. Moreira, L. F. Rocha, G. Veiga and E. Moreira, "Evaluation of Depth Sensors for Robotic Applications," 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, Vila Real, 2015, pp. 139-143.

[11] S. Shen, N. Michael and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, 2012, pp. 9-15.

[12] Puričer, P. - Kovář, P.: Technical Limitations of GNSS Receivers in Indoor Positioning. In Proceedings of 17th International Conference Radioelektronika 2007. Brno: FEKT VUT v Brně, 2007, p. 205-209. ISBN 1-4244-0821-0.