Autonomous Navigation and SLAM on a Hexapod

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Parva Patel
ID No. 2014A3TS0334G

Under the supervision of:

Prof. Howie Choset
&
Guillaume Sartoretti

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS
December 2017

Certificate

This is to certify that the thesis entitled, "*Autonomous Navigation and SLAM on a Hexapod*" and submitted by Parva PATEL ID No. 2014A3TS0334G in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

[illegible]

Supervisor
Prof. Howie CHOSET
Professor,
Carnegie Mellon University
Date:

[illegible]

Co-Supervisor
Guillaume SARTORETTI
Postdoctoral Fellow,
Carnegie Mellon University
Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

Abstract

Bachelor of Engineering (Hons.) Electrical and Electronics Engineering

Autonomous Navigation and SLAM on a Hexapod

by Parva Patel

The project aims on making a generalised Autonomous Navigation and Mapping system to explore unknown area and traverse through known environments with limited sensing capabilities, motion primitives and computational power.The system is designed in such a way that it can be integrated with and deployed on any ROS based platform.

Acknowledgements

I would like to acknowledge the continuous support of my parents throughout my undergraduate thesis and in fact, my entire academic life. Thanks to Professor Howie Choset for being such a helpful advisor and giving constructive feedback all along.

I would also like to mention Guillaume Sartoretti, Hadi Salman and Lu Li whose guidance and inspiration all throughout the 6 months at The Robotics Institute of Carnegie Mellon University made it possible for me to complete this thesis. I am also quite grateful to my on-campus advisor Prof. Sarang Dhongdi for all the help.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture

Contents

Chapter 1

Introduction

The "snake monster" of the biorobotics lab is a six-legged robot which is capable of traversing complex and rough terrains. The goal of this project is to make it autonomously navigate through known/unknown environments such as hallways of a building while creating a map and simultaneously localising itself in the map.

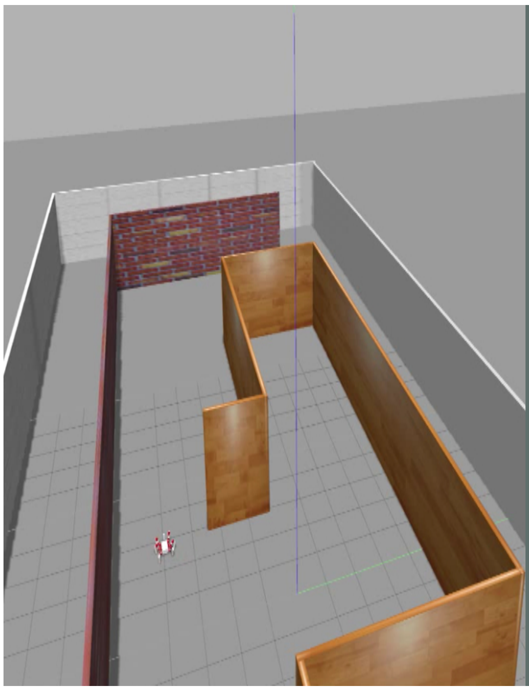


Figure 1.1: Environment in simulator

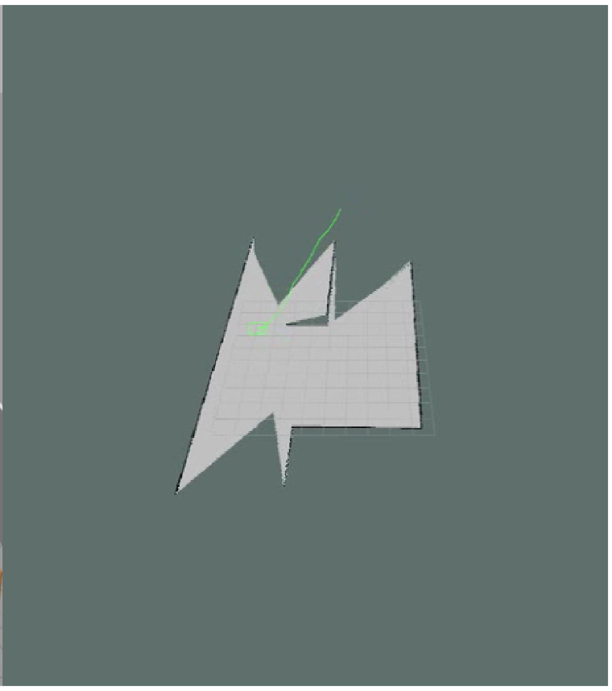


Figure 1.2: Visualisation of Mapping process

For an unknown environment, the robot should be able to build a partial map from its current position and localise itself in the map. At the same time, the robot should also figure out the next best position in the partial map to go and scan to continue the mapping process. Once the robot has figured out the next best position, it should be able to navigate to that position using the partial map.

In order to move to the next best position, the robot has to make a global plan to reach that position and navigate there by avoiding obstacles by relying on the current partial map.

Chapter 2

Related Work

2.1 Hexapod

A hexapod is a six-legged walking robot. The robot used for this project, named “Snake Monster”[11], has 3 joints on each of the six legs. It is a completely modular robot[6] and uses modules from ‘Hebi Robotics’. It has been successfully tested over various rough and rocky terrains for its robustness.

It uses an algorithm called Central Pattern Generator(CPG)[9] for generating the gait. The CPG algorithm[5] takes higher level commands from the user and converts them into angles for all 18 joints.

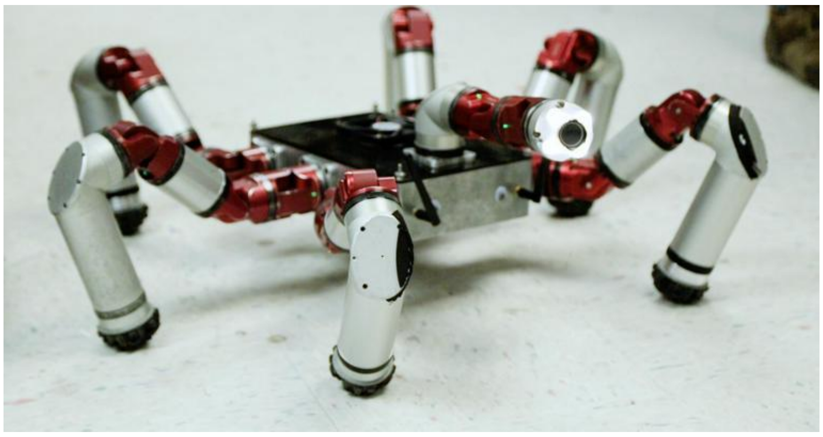


FIGURE 2.1: Snake Monster (source: biorobotics.ri.cmu.edu)

2.2 SLAM

Simultaneous Localisation and Mapping[2] has always been a very intriguing topic in the field of mobile robots. It is about making the robot capable of mapping an environment and at the same time, estimating it's position in the environment. There have been various approaches[3] in doing so, using different types of sensor configurations.

Most of the SLAM algorithms use vision based systems like stereo cameras along with range sensors for depth perception to create a point cloud of the environment for mapping purposes. They use odometry data from sources like wheel encoders or motion capture systems enhanced with data from onboard IMUs to localise the robot in the map.

2.3 ROS Navigation Stack

ROS Navigation stack consists of a group of packages organised in a manner as to allow robots to navigate through a pre-build static 2-D Map. It contains the following:

- **Move base package:** A package to control the mobile robot which can covert messages of type"path" from the path planner into velocity commands ("Twist" message).
- **Costmap 2d:** A package to enable costmaps in a given global frame publish them as an obstacle occupancy grid over a static 2-D Map.
- **Action Library Client:** A client that can fetch a 2-D Navigation goal from Rviz and send it to the path planner as a "Pose" message.

- **Global Planner:** A path planner to plan a path in the global frame i.e. the world fixed frame to the navigation goal. The default global planner for Navigation stack is "NavFnROS planner".
- **Local Planner:** A planner that used Laserscans from the Lidar to avoid dynamic obstacles while allowing the robot to follow the path planned by Global Planner. The default local planner for ROS is "TrajectoryPlannerROS".

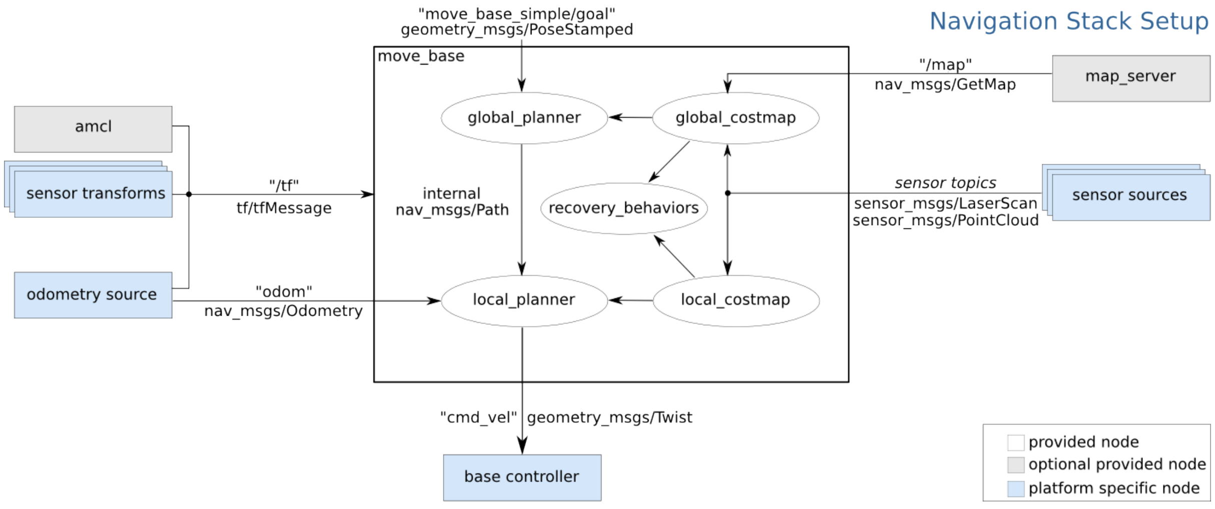


FIGURE 2.2: Inflation (source: http://wiki.ros.org/navigation/Tutorials/RobotSetup)

Chapter 3

Mission

The mission of this project is to create an Autonomous Exploration system on ROS to explore unknown environments and navigate through them. This will be achieved using minimal sensing abilities, computing power and motion primitives. The system is designed such that it can be used with any pre-existing robots having a Lidar and an odometry source working on ROS.

Challenges:

The robot being used as a platform to create this exploration system is a hexapod. Being a legged robot, it has fewer motion primitives and less agility as compared to mobile robots with wheels.

We will use only Lidar and on Odometry sensor for localisation, exploration and navigation. The algorithm doesn't require any visual sources like cameras and hence, no image processing is required, which significantly reduces the computational complexity. This makes the system easily integrable with on-board processors.

Chapter 4

Sensor system

The sensor system designed to perform this task consists of a LiDAR and an odometry sensor.

4.1 LiDAR

The LiDAR used for this project is Hokuyo URG-04LX-UG01 Lidar. It is a 2D short range LiDAR with a maximum range of 5.6 meters, an angular resolution of 0.25 degrees and an angular range of 240 degrees.

A LiDAR provides us with a 2D point cloud with the distance of each point in its range. This data is used to build a map and avoid obstacles in the path.

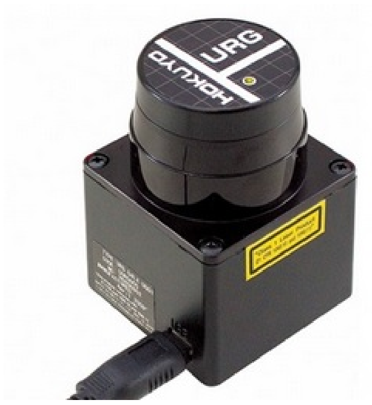


FIGURE 4.1: Hokuyo URG-04LX-UG01 Lidar

4.2 Odometry Sensor

The odometry sensor used for this project is the VOOF camera sensor which consists of two 6000 fps cameras and an IMU.

The odometry data is used to estimate the position of the robot in the map.

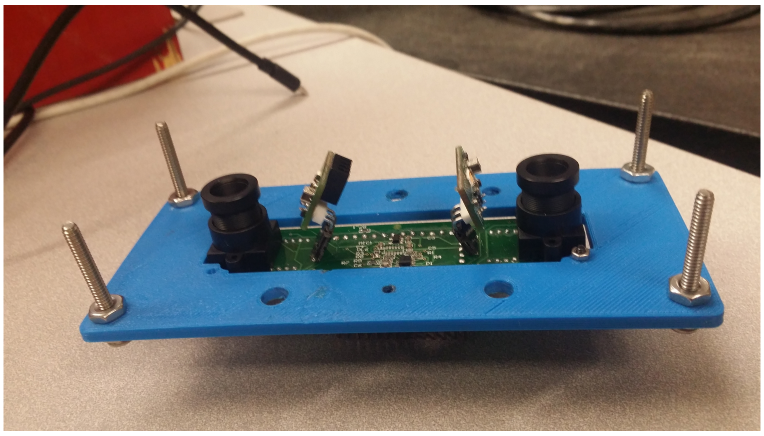


Figure 4.2: VOOF Sensor with mount

Chapter 5

Software Architecture

The Robot Operating system (ROS)[8] is used for this project. It is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.

5.1 Frames

ROS uses a set of rules to define the structure of a robot. Each part of the robot is assigned a coordinate frame. Coordinate systems in ROS are always in 3D, and are right-handed, with X forward, Y left, and Z up. A frame is represented by it's pose in a 3-D environment. The standard structure of representing a pose is:

Point (Origin) in [x,y,z]

Orientation in [x,y,z,w] (Quaternion)

5.2 Transforms

The relationship between two frames is represented by a 6 DOF relative pose called a Transform. A transform contains a translation followed by a rotation. The standard structure of a transform is:

Translation in [x,y,z]

Rotation in [x,y,z,w] (Quaternion)

If the relative position of two frames doesn't change over time, the transform between them is a static transform.

5.3 Structure of our Robot

We are following a standard structure defined for mobile robots which consists of the following frames:

1. **Map :**

   The map frame is a world fixed frame and is used for long-term global referencing.

2. **Odom :**

   Odom is also a world fixed frame if the odometry source is perfect. Inacccurate odometry sources require drift correction, making odom frame to be dynamic with respect to the map frame. This dynamic transform is provided by the localisation algorithm.

   In our setup, the odom frame is calculated by using the transform between odom and base_link which is provided by the odometry sensor.

The position of the robot is supposed to be continuous over time in the odom frame and hence it is an accurate frame for local referencing, but drift makes it a poor frame for long-term referencing.

3. **Base_link :**

   The frame base_link represents the base of the robot. It is used for getting the position of the robot. The base_link frame here is treated at a footprint of the base on the ground (sometimes also referred to as base footprint frame). As it is a footprint on the ground, the roll and pitch of the frame always stays constant and zero with respect to the global frame map.

4. **Laser_link :**

   The frame laser_link is attached to the frame base_link and represents the position of the Laser Range sensor in the world.

   For our setup, this Laser Range sensor is rigidly attached to the base of the robot and hence, the transform between laser_link and base_link is a static transform.

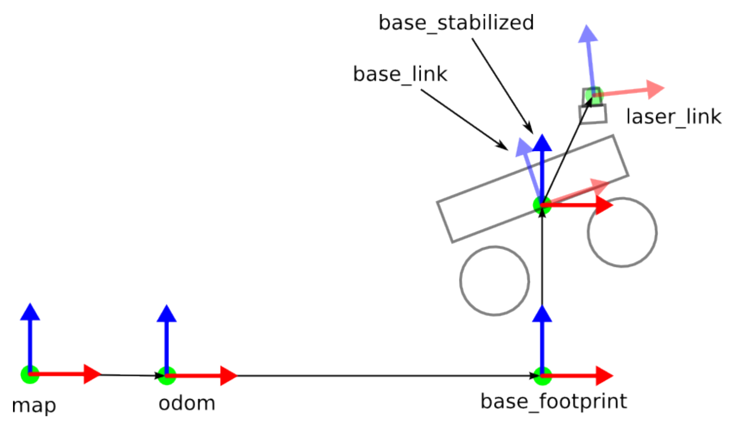


FIGURE 5.1: Setup for Hector SLAM (source: http://wiki.ros.org/hector_slam)

Chapter 6

Mapping

6.1 Static Map

Hector slam[7] is the algorithm being used to map the environment using the LiDAR and the Odometry sensor. The reason behind selecting hector slam is that it provides 2D pose estimates at scan rate of the sensors and works well with low range LiDARs in an environment with very high amount of features available for detection.

To close the loop and provide hector mapping algorithm with more accurate position of the robot, odometry data is added to produce a dynamic transform between a world fixed frame (map frame) and the robot frame using gmapping [4] (a ROS package). The dynamic transform compensates for the drift of the odometry sensor.

6.2 Costmaps

The costmap is a data structure that represents places that are safe for the robot to be in a grid of cells. The values in the costmap are binary, representing free space and locations where there is a likelihood of collision.

A costmap broadly classifies the space into 3 categories:

1. **Free Space**

   The space where a robot can traverse freely and there is absolutely no risk of a collision.

2. **Occupied Space**

   The space where a robot is most likely to face a collision if it tries to cross that area. That space is therefore out of bounds for traversal.

3. **Unknown Space**

   This is the space that has not been added to the global costmap yet. It has not been determined whether this space is safe to traverse or not.

6.3 Inflation

Inflation is the process of propagating cost values out from occupied cells that decrease with distance. To do this, we define 5 specific states for costmap values as they relate to a robot.

- **Lethal:** A cell in the occupancy is given lethal status when it contains an actual obstacle.
- **Inscribed:** A cell with Inscribed status means that it is near to the cell with lethal status. The distance between inscribed and lethal cell is less than the size of the robot and hence, robot will surely collide if it's center enters the Inscribed cell.
- **Possibly circumscribed:** The cell with this status is almost similar to the cell with Inscribed status. The only difference is that instead using robot's size as a threshold to predict collision, this uses robot's

orientation to predict collision. If the robot's center lies in this cell, then it depends on the orientation of the robot whether it collides with an obstacle or not.

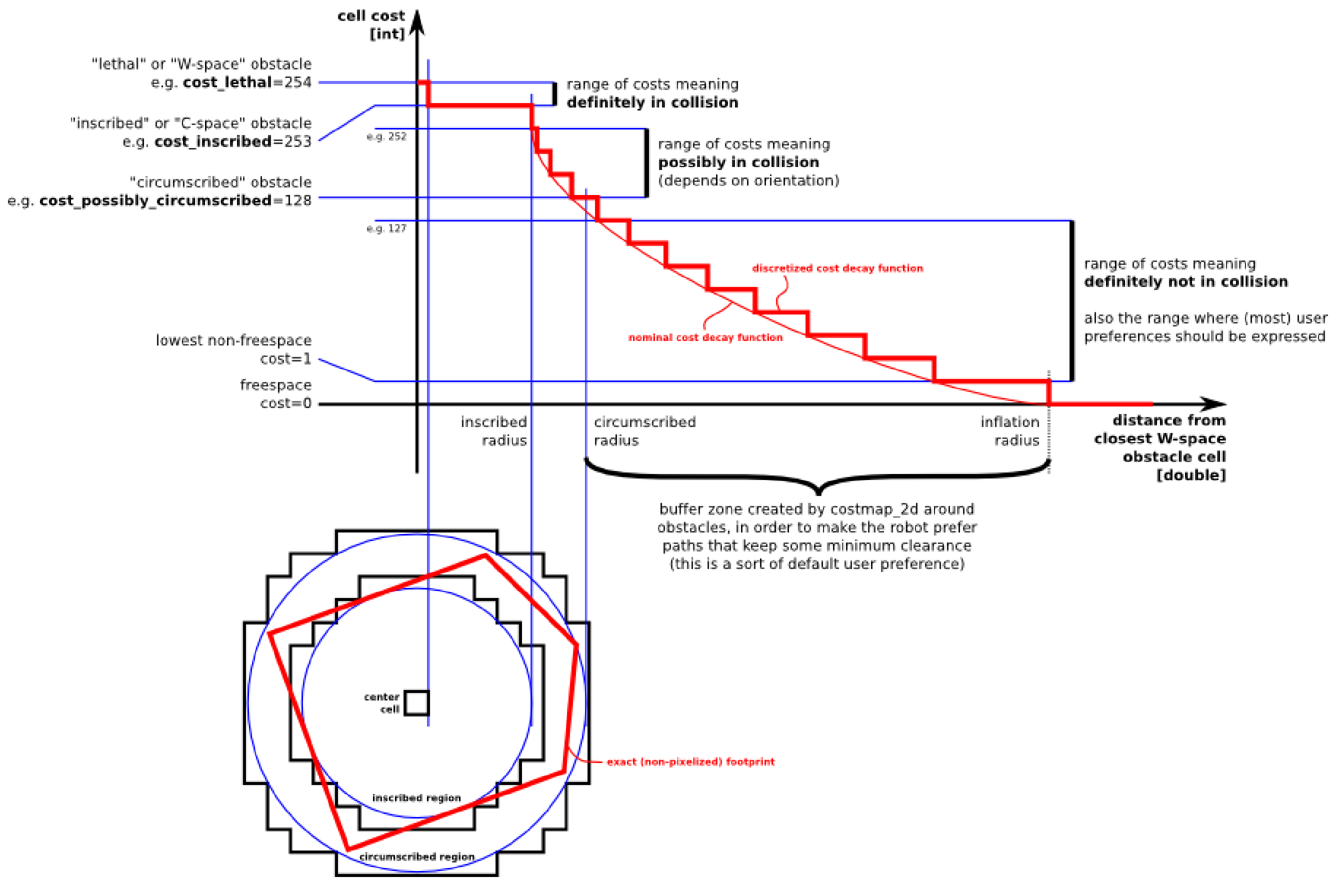

cell cost [int]
"lethal" or "W-space" obstacle e.g. cost_lethal=254
"inscribed" or "C-space" obstacle e.g. cost_inscribed=253
"circumscribed" obstacle e.g. cost_possibly_circumscribed=128
lowest non-freespace cost=1
freespace cost=0
e.g. 252
e.g. 127
range of costs meaning definitely in collision
range of costs meaning possibly in collision (depends on orientation)
range of costs meaning definitely not in collision
also the range where (most) user preferences should be expressed
discretized cost decay function
nominal cost decay function
inscribed radius
circumscribed radius
inflation radius
distance from closest W-space obstacle cell [double]
buffer zone created by costmap_2d around obstacles, in order to make the robot prefer paths that keep some minimum clearance (this is a sort of default user preference)
center cell
inscribed region
circumscribed region
exact (non-pixelized) footprint

FIGURE 6.1: Inflation (source: http://wiki.ros.org/costmap_2d)

- **Freespace:** Cells with this status are assumed to be flat ground with no risk of collision. The robot can traverse freely in this space.
- **Unknown:** This status represents a cell about which the algorithm has no knowledge. All the cells in area which is not mapped or is yet to be mapped are given unknown status.
- All other cells are assigned a value between "Freespace" and "Possibly circumscribed" depending on their distance from a "Lethal" cell and the decay function provided by the user.

Chapter 7

Localisation

Adaptive Monte Carlo localization (AMCL)[1]package in ROS, also known as particle filter localization is being used for localisation in this project. It is an algorithm for robots to localise using particle filter. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves around and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e., a hypothesis of where the robot is. The algorithm typically starts with a uniform random distribution of particles over the configuration space, the robot assumes it is equally likely to be everywhere as it has no prior information regarding the position. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayern estimation i.e., how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot.

Chapter 8

Navigation

8.1 Algorithm

The project is about allowing snake monster to autonomously explore and map its environment, using onboard sensing. Mapping and localisation (SLAM) are the essential parts of this system. After building a partial map from its current position, the robot should be able to figure out a position in the map from which it will get its next best view to continue the mapping process. This is basically done by choosing the point which would decrease the entropy of the map the most. Once it figures out the position with next best view, it should be able to navigate itself to that position using the map built from the current position while avoiding obstacles in its path. Navigation stack from ROS was used for this setup.

8.2 Assumptions

In the beginning, the robot makes 2 assumptions:

- Its starting position is the origin (0,0) of the 2-D Map.
- It is currently in a free space with no obstacles in its vicinity.

8.3 Exploration

8.3.1 User Controlled

The robot now needs a goal position from the user, which it will try to achieve using the shortest path possible and also at the same time, map the environment while traversing to that position. It uses A* algorithm[10] for planning the path to the given position.

8.3.2 Frontier Exploration

To make the SLAM completely autonomous, an algorithm called frontier exploration[12] was used. It processes the map built from the initial or current position and extracts the frontiers of the map. One of the frontier, based on size, is selected as a goal position to further explore the area.

8.4 Navigating to the Goal

Once it gets a goal position, it plans a straight line path towards that position as it has assumed that it is in a free space. The gait generated by the CPG enables the robot to follow that path. While following the path, it constantly keeps on updating the occupancy grid by updating the local and global costmaps using laserscans from the Lidar.

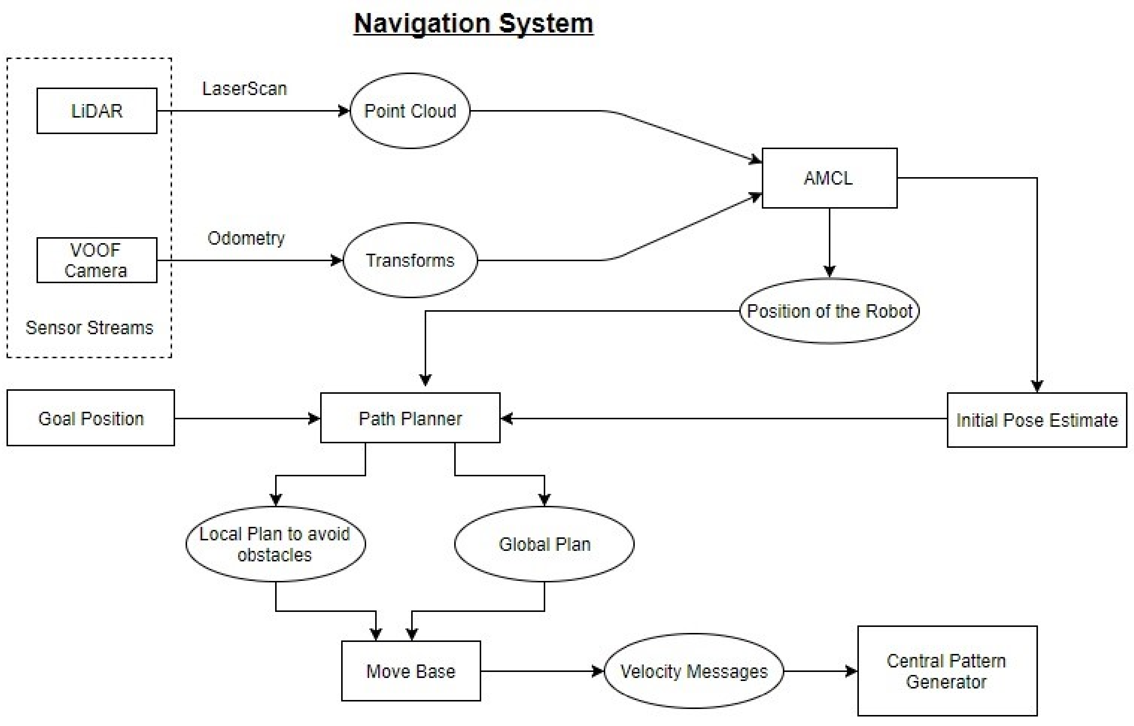
While doing so, if it finds that it might encounter an obstacle while following the current path, it immediately cancels the current plan and re-plans a new path to achieve the goal using A* algorithm. The new plan is such that it avoids the obstacle that led to failure of the previous plan.

A mapping node is constantly running while it is traversing to the goal position enabling it to create a 2-D map of the environment.

Chapter 9

ROS Structure

The ROS structure consists of nodes for taking in sensor data, building a map, localisation, path planning and central pattern generation for the snake monster. Everything works in a specific sequence as shown in the flow chart.



Chapter 10

Conclusion

The Final result of the project is an autonomous system for Exploring unknown environments and navigating through known spaces in the form of a ROS package. The system is designed such that it overcomes the challenge of limited motion primitives in the case of legged robots, and also at the same time, takes full advantage of agility if applied on mobile robots. It is easily integrable with any existing platform working on ROS having a Laser Range sensor and any odometry source.

Here is a YouTube video of the hexapod exploring unknown environment in gazebo simulator: `https://www.youtube.com/watch?v=q6TBETfsYtA`

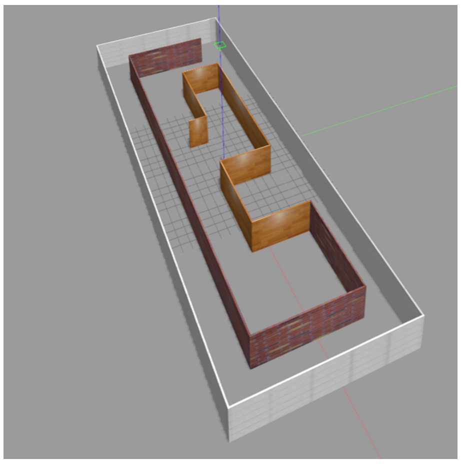


FIGURE 10.1: Arena used for testing

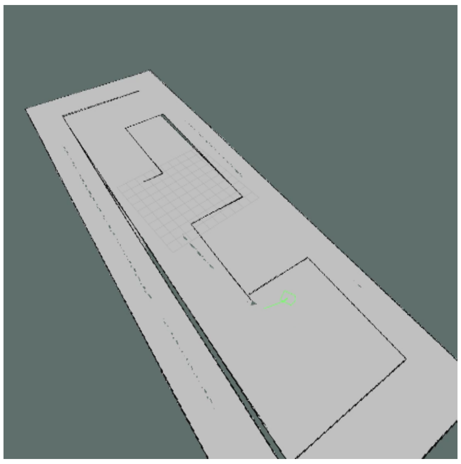


FIGURE 10.2: Complete map of the Arena

Bibliography

[1] Frank Dellaert et al. “Monte carlo localization for mobile robots”. In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on.* Vol. 2. IEEE. 1999, pp. 1322–1328.

[2] MWM Gamini Dissanayake et al. “A solution to the simultaneous localization and map building (SLAM) problem”. In: *IEEE Transactions on robotics and automation* 17.3 (2001), pp. 229–241.

[3] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[4] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved techniques for grid mapping with rao-blackwellized particle filters”. In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46.

[5] Auke Jan Ijspeert. “Central pattern generators for locomotion control in animals and robots: a review”. In: *Neural networks* 21.4 (2008), pp. 642–653.

[6] Simon Kalouche, David Rollinson, and Howie Choset. “Modularity for maximum mobility and manipulation: Control of a reconfigurable legged robot with series-elastic actuators”. In: *Safety, Security, and Rescue Robotics (SSRR), 2015 IEEE International Symposium on.* IEEE. 2015, pp. 1–8.

[7] Stefan Kohlbrecher et al. “Hector open source modules for autonomous mapping and navigation with rescue robots”. In: *Robot Soccer World Cup.* Springer. 2013, pp. 624–631.

[8] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software.* Vol. 3. 3.2. Kobe. 2009, p. 5.

[9] Ludovic Righetti and Auke Jan Ijspeert. “Pattern generators with sensory feedback for the control of quadruped locomotion”. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.* IEEE. 2008, pp. 819–824.

[10] Anthony Stentz. “Optimal and efficient path planning for partially-known environments”. In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on.* IEEE. 1994, pp. 3310–3317.

[11] Matthew J Travers et al. “Shape-Based Compliance in Locomotion.” In: *Robotics: Science and Systems*. 2016.

[12] Brian Yamauchi. “A frontier-based approach for autonomous exploration”. In: *Computational Intelligence in Robotics and Automation, 1997. CIRA’97., Proceedings., 1997 IEEE International Symposium on*. IEEE. 1997, pp. 146–151.