# Intelligent Obstacle Avoidance for Hyper-redundant Robots

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

by

**Perugu Bhuvanna Chaitanya Reddy**
**(Roll No. 130108001)**

Under the guidance of

**Dr. Prithwijit Guha**
**Dr. Srinivasan Krishnaswamy**


DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
April 2016

# CERTIFICATE

This is to certify that the work contained in this thesis entitled

**Intelligent Obstacle Avoidance for Hyper-redundant Robots**

is the work of

**Perugu Bhuvanna Chaitanya Reddy**
(Roll No. 130108001)

for the award of the degree of Bachelor of Technology, carried out in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

_____

Guide

Date: _____

Place: _____

# DECLARATION

The work contained in this thesis is our own work under the supervision of the guides. We have read and understood the "B. Tech./B. Des. Ordinances and Regulations" of IIT Guwahati and the "FAQ Document on Academic Malpractice and Plagiarism" of EEE Department of IIT Guwahati. To the Best of our knowledge, this thesis is an honest representation of our work.

_____

Author

Date: _____

Place: _____

# Abstract

Robotics has come a long way since it stepped out of the pages of Isaac Asimovs science fiction novels into the real world in the mid $20^{th}$ century. As technology progressed, there rose a need for automated machines capable of performing tasks beyond the capabilities of a human. In the modern world, robots of all shapes and sizes are being studied and developed for various applications. One such family of robots is the hyper-redundant manipulators. Although not as widespread as the usual mobile robots, these robots have a specific set of properties which make them the best option for deployment in unfriendly terrains and for multi-functional applications like outer-space, military and search-rescue missions.

The fundamental goal of this thesis project is to further the research of hyper-redundant robots, especially the problem of navigation in unpredictable terrains. We define the problem statement as follows: **given a planar snake robot, design an architecture for autonomous navigation of the robot in unpredictable environments**. This project targets the ambitious objective of exploring obstacle avoidance algorithms of varying intelligence.

In this documentation, we discuss, at length, how each half of the problem statement is separately developed to complement one another and then fit together to form the complete system. The first semester of the project concentrated on the robotics aspect, i.e. simulation, locomotion and perception of the snake robot. In analogy terms, the car was manufactured. The second semester focused on developing obstacle avoidance algorithms. To continue the analogy, the driver of the car has been designed/trained. In the end, we attempted to implement our methods on a real robot and validate the simulation results and algorithm's performance.

# Contents

# List of Figures

# Chapter 1

# Introduction

This chapter talks in length about some of the concepts which form the foundation of this thesis project. We start by discussing hyper-redundant robots and then move on to navigation and obstacle avoidance.

## 1.1 Hyper-redundant robots

While robotics has its origins in the early $20^{th}$ century, hyper-redundancy in robots was not conceived until about 1970. Redundancy in robots is defined as the property of having more degrees of freedom than what is necessary for the robot application. Consider the case of a robot operating in 3D space. The three translational axes and three rotational axes make up six degrees of freedom in total. Hence, any robot with seven or more degrees of freedom would be termed kinematically redundant. The word hyper-redundancy implies redundancy to a very large extent. One of the best examples of hyper-redundancy in nature are snakes which served as the original inspiration for the concept.

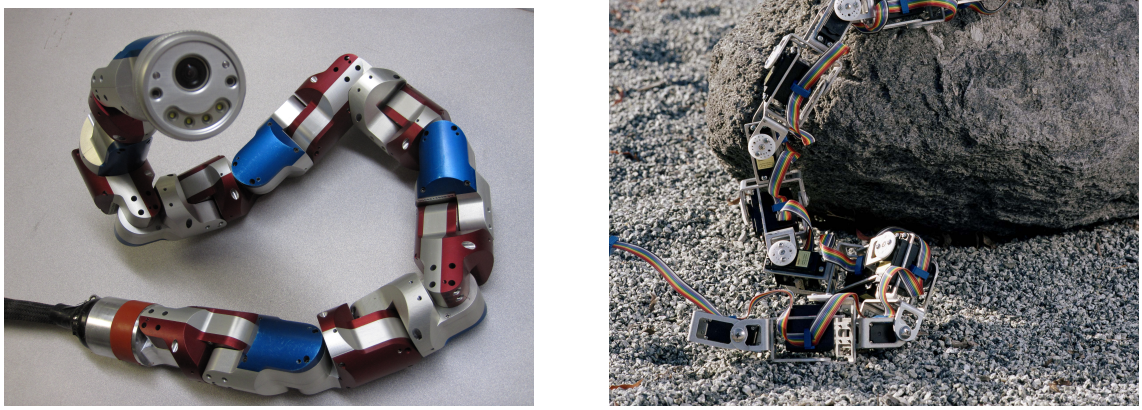

Figure 1.1: Snake Robots by CMU and NASA

While hyper-redundancy in robots may seem like a wasteful endeavor at first sight, a closer look reveals the array of advantages this approach brings. Such a property enables the robot to assume odd geometries which help them navigate highly constrained environments like tunnels, pipes and tight spaces. Their modular morphology allows complicated trajectories and makes them resistant to partial failures. The same robot can climb a tree, swim in water and climb stairs with no physical modifications. Clearly, hyper-redundant robots are the Swiss Army Knife of robots.

Yet, snake robots have been restricted to research laboratories due to the challenges such a configuration brings with it. Conventional methods of kinematic modeling have not been particularly successful for hyper-redundant manipulators. Their inherent structure makes it challenging to design and program efficient control algorithms. More research into these interesting machines will help discover a path to bringing them into the practical world.

## 1.2 Navigation & Obstacle Avoidance

Navigation refers to the ability of a mobile robot to travel to a destination in an environment, known or unknown, without colliding into obstacles. It is one of the fundamental tasks a mobile robot should be able to perform, especially if it is to be called autonomous. It is not to be confused with the 'gait' of a robot, which refers to the way the robot moves its joints in order to travel forward. Navigation requires the robot to have some idea about its surroundings so that the robot can control its heading in real-time and avoid obstacles. This can be achieved in a variety of ways such as an RGB camera vision, laser scan mapping or ultrasound echolocation.

In this thesis project, we choose a snake robot as our hyper-redundant manipulator. The snake robot has a wide range of bio-inspired gaits to choose from but we go with the most common, 'lateral undulation'. Lateral Undulation consists of propagating sinusoidal waves back through the length of its body, resulting in a smooth forward motion, akin to the real snakes. With mounted ultrasonic sensors, the robot builds a rudimentary map of its surroundings which is then given to avoidance algorithms to make a decision about the heading of the snake.

# Chapter 2

# Literature Review

Snake robots have been an object of interest for researchers since the advent of hyper-redundant robots. Hirose, a pioneer in the field, modeled the motion of biological snakes as a mathematical equation, now known as the 'serpenoid curve'. His ACM robot series inspired a generation of snake-robot researchers who built massively upon his work. Chirikjian further developed the kinematics of snake robots, characterizing the gaits in terms of the backbone curves. Multiple researchers have studied the controllability and dynamics of these serpentine systems and proposed controllers for the same. Liljeback's 'Snake Robots: Modelling, Mechatronics, and Control' is one of the most comprehensive books on the topic.

The interest in such robots has since then exploded, with most of the research being done in Japan and USA. Mark Yim of the University of Pennsylvania started his research into snake robots with his dissertation at Stanford, in which he discusses the various gaits of such systems. Howie Choset of Carnegie Mellon University manufactured a versatile snake robot at his lab, complete with implementations of unconventional gaits like rolling, sidewinding and swimming. Research teams at Johns Hopkins University developed robotic snakes for medical purposes. New research is being done towards the development of fish and eel-like robots which have similar mechanisms under the hood.

The majority of the research focus is on improving the control schemes for redundant robots. This leaves space for work to be done in the navigation of existing snake robots. Reliable navigation and obstacle avoidance is an important aspect for any robot and that is what we

intend to focus on in this project.

# Chapter 3

# Previous Work

## 3.1 Simulation of the Snake Robot

Robots are costly machines to maintain and operate. Hence, robotics researchers prefer to test their algorithms on robot simulations as this allows for greater freedom to experiment with infeasible scenarios with zero consequences. Among the variety of robotics simulators available in the market, we chose the Gazebo simulator with ROS middle-ware.

For the scope of this thesis project, we use a planar snake robot with passive wheels and servo-actuated joints between links. The design is based on a prototype robot built as part of the sixth semester Design Project under Dr. Prithwijit Guha. Since the robot is a combination of modules, we first create a 3D model of a single module. It is made of a 'Z' shaped metal link, cut at one corner joint to make space for the wheel. A free-rotating wheel is mounted on an axle and fixed at the cut corner joint. A servo is attached to the higher step of the metal link such that it can be rigidly fixed to the lower step of the next link. For this simulation, we use a black cylinder to represent this servo motor. The two movable joints of the module are modeled as a full rotation 'continuous joint' (for the passive wheel) and a limited rotation 'revolute joint' (for the servo motor).

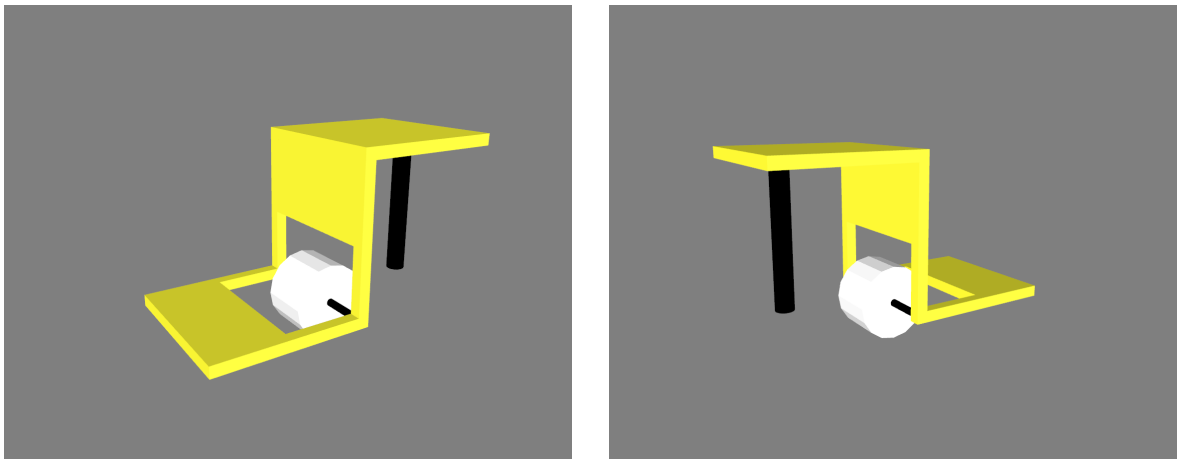

Figure 3.1: 3D rendering of the snake module

Good simulation behavior requires careful modeling of the dynamic, kinematic, interaction and surface parameters of the real robot. For 3D simulation of the whole snake robot, we create multiple instances of the module and link them with the servo motors. The wheels are given anisotropic friction coefficients and each part of the robot is given a mass and inertia for proper rendering and physics computations in Gazebo.

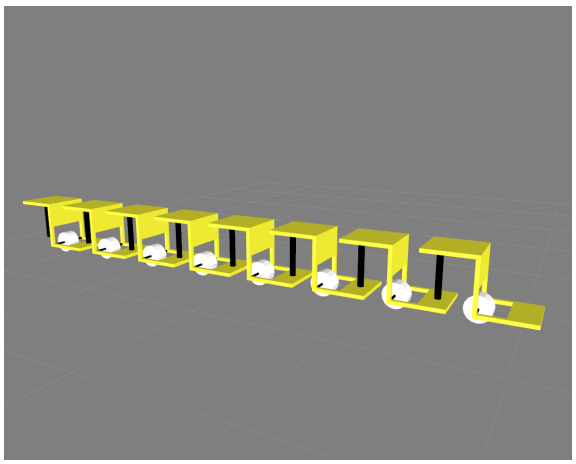

Figure 3.2: 3D rendering of the complete snake robot

Gazebo simulations use URDF or SDF files which are a link-by-link description of the complete robot model in XML. For complicated robots such as ours, these URDF files can reach thousands of lines of code, which is why we make use of Xacro for simplifying large XML documents.

### Robot Operating System (ROS)

ROS is a suite of software frameworks, libraries and packages providing extensive software implementations and support for robot simulation and control. Despite not being a real operating system, it provides OS-like features and functionalities and is extremely popular among the robotics research community.

### Gazebo Simulator

Gazebo is a popular 3D robotics simulator with a very rich feature array. It comes bundled with ROS, making it the best choice for ROS users. It employs four physics engines (ODE, Bullet, Simbody, DART) for real-world robot simulation and a rendering engine (OGRE) for 3D rendering of URDF/SDF files.

## 3.2 Gaits, Locomotion & Teleoperation

Snake robots rarely used actuated wheels to move around in the world as this body morphology is not conducive for such drive mechanisms. Instead, researchers have drawn inspiration from biological snakes and used the robots many degrees of freedom to implement generalized gaits for locomotion in various kinds of environments. Some of the well-known ones are lateral undulation, sidewinding, concertina, sinus lifting, and swimming. Such gaits are known as parameterized gaits as they can be described by an underlying equation and a set of parameter values. There are more complex gaits called scripted gaits which are designed specifically for a particular task. Since, our focus is on the navigation aspect of the robot, these complex gaits are out of the scope of this project.

We adopt one of the most commonly used parameterized gait, *'the lateral undulation'*. As already discussed, the essence of this gait is propagating sinusoidal waves back into this body, generating a net forward push from the asymmetrical friction of the ground. Hirose's 'serpenoid curve' describes the angular movements of the joints between the links of the snake robot, as a function of time and distance from head.

$$\phi_{i,t} = \alpha sin(-\omega t + \delta_i) + \phi_0 \tag{3.1}$$

Here, $\phi_{i,t}$ is the angular displacement of joint I at time t. The parameters of this equation are $\alpha$, which determines the amplitude of the oscillation in the serpentine body, $\omega$, which decides the angular frequency of the sine waves, $\delta_i$ is the offset of each joint due to its displacement from the head of the robot and $\phi_0$ refers to the rotation angle which changes the robot's heading.

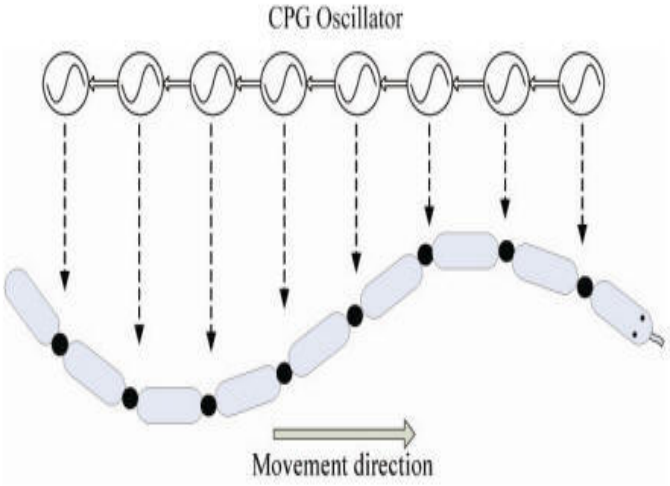


Figure 3.3: Representation of lateral undulation

For our 8-link robot, there are 7 actuated joints that have to be controlled to execute lateral undulation. We set $\delta_i$ to $\pi/4$ and create a Central Pattern Generator (CPG) for each joint which feeds it input according to Hiroses equation. The parameters of the equation play an important role in the real-time locomotion of the robot. $\omega$ controls the speed of forward motion while $\delta$ decides the number of sine waves along the length of the snake robot. The value of $\phi_0$ steers the robot, with positive values driving it right and negatives values leading to a left turn. $\alpha$ has to be chosen such that no single servo experiences a big angular input.

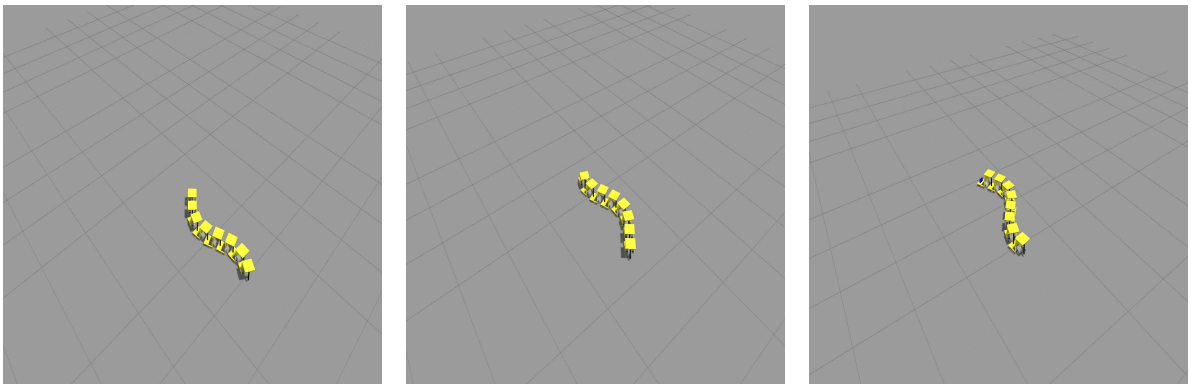

Figure 3.4: Simulated snake robot performing lateral undulation

Teleoperation, in general, refers to a system of controlling robots from a distance. To demonstrate the versatility of the Hirose's serpentine equation, we built a teleoperation module for the simulated snake robot which is controlled by the 'WASD' keys on the keyboard. The concept is simple: 'W' increases the speed of the robot while also redirecting it to go straight, while 'S' reduces the speed, possibly below zero, making the snake robot crawl backwards. The 'A' key makes the robot make continuous left turns and the 'D' key does the same to the right side.

## 3.3 Perception of Surroundings

As discussed earlier, autonomous navigation requires the robot to have at least a rudimentary idea of its surroundings based on which it can take decisions about its heading. Hence, the robot has to have an on-board sensor which can return information about the surroundings and help maintain a map of nearby obstacles. There are a plethora of options available for such sensors but we will opt for the *'Hokuyo Laser Scanner'* which returns a scan map of the distance to obstacles in its field of view. Gazebo contains a pre-built plugin which simulates this scanner.

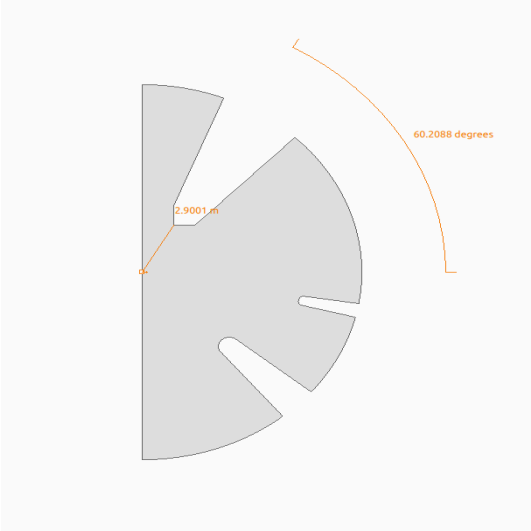


Figure 3.5: Laser Scan visualization in Gazebo

The issue that arises with this configuration is that, due to the nature of lateral undulation, every module of the snake robot constantly swings from one direction to the other. As a result, the sensor data, obtained in real-time, also oscillates along with the module it is mounted on. One of the unconventional configurations we are considering is one which has an ultrasonic sensor mounted on both sides of every module. This might help keep everything in the line of sight and can be synchronized using the known relationship of lateral undulation.

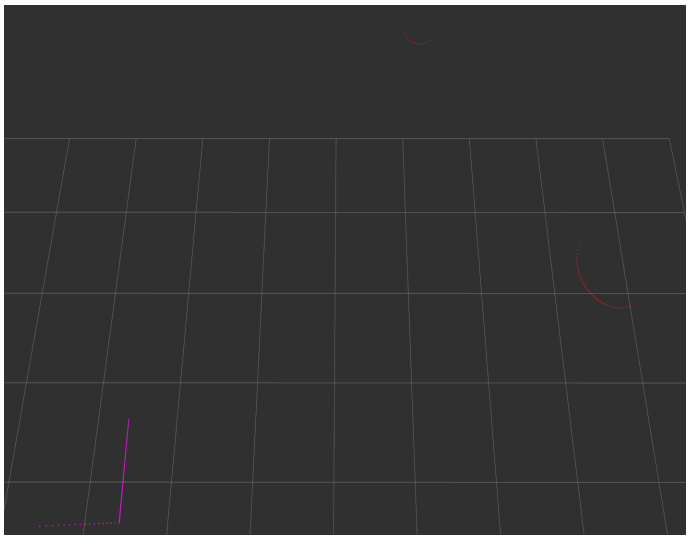

Figure 3.6: Laser Scan visualization in RViz

Each ultrasonic sensor is replicated in simulation by a Hokuyo laser scanner of a very small angular range. The average of the laser scan array is equivalent to the ultrasonic sensor array.

# Chapter 4

# Obstacle Avoidance

We now turn our attention towards the obstacle avoidance problem in hyper-redundant robots. The high complexity of snake robots is evident from the previous discussion, but Hirose's 'serpenoid curve' simplifies the kinematic control of these robots to a large degree. Instead of handling the multiple degrees of freedom at once, we can now control the speed and direction of motion with a few parameters ($\omega$ and $\phi_0$). Since the goal is to avoid obstacles, we will concern ourselves with just the heading ($\phi_0$) of the robot.

In this chapter, we discuss some algorithms which enable to a laterally undulating snake-robot to sense and dodge obstacles in its vicinity. The robot's awareness of its surroundings comes solely from the ultrasonic sensors mounted on the sides of its modules. This sensor array returns a representation of the world around it, based on which our algorithms take action by controlling the heading ($\phi_0$) variable.

In a sequence similar to evolution, we first start with a rudimentary method and slowly impart intelligence into the algorithms to make them better and better at the task at hand.

## 4.1 Blind Avoidance

This section talks about a rudimentary thumb-rule based algorithm for a snake-robot with mounted ultrasonic sensors to move away from detected obstacles. It is similar to the tech-

nique used by organisms at the initial stages of evolution. In essence, the algorithm measures the relative safety of its surroundings in real-time and moves towards the region of higher safety.

The algorithm roughly estimates the direction of the approaching obstacle from considering the sensor outputs directionally and makes a turn towards the opposite direction depending on how close the obstacles is. The term 'blind' in the title refers to the algorithm's inability to plan its actions due to lack of sufficient information beforehand.

### 4.1.1 Hardware

The actual robot used for testing these algorithms is a modified version of the prototype created as a sixth semester Design Project under Dr. Prithwijit Guha. As summarized earlier, the robot is made of 8 steel links connected together by 7 servo motors. It is powered by an external source and uses an on-board Arduino as the control unit. The robot has also been fitted with an array of 16 ultrasonic sensor on both sides of the robot which feed information to the on-board Arduino.

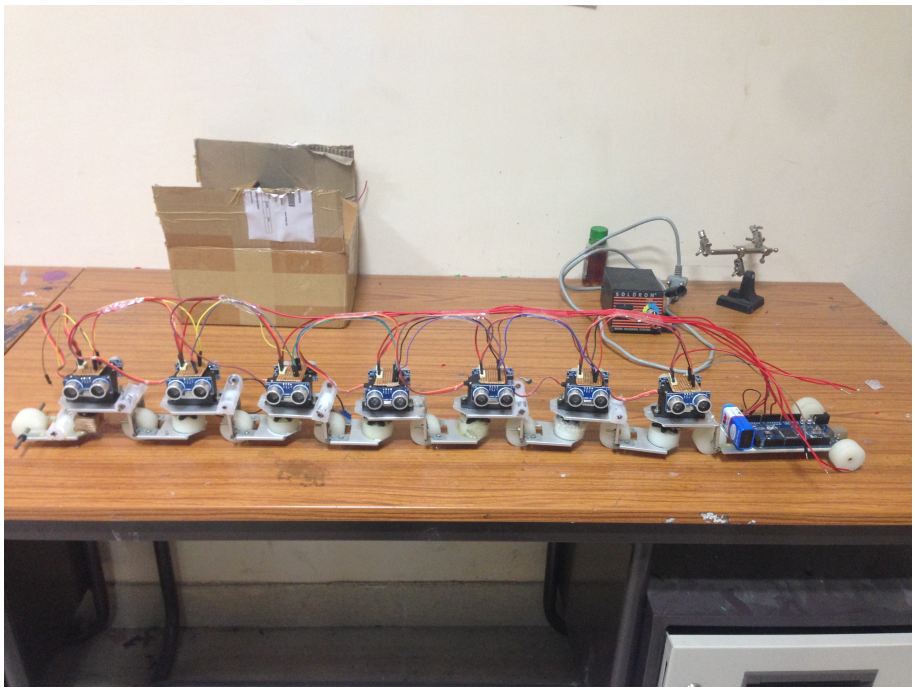

Figure 4.1: Construction of snake robot hardware

### 4.1.2 Algorithm

The sensor array has been mounted such that the outputs of the left-facing sensors represents the world to the left of the robot while the right-facing ones describe the world to the right of the robot. Independently computing the average of the left and right sensor outputs condenses the complete world information into just two numbers.

A obstacle to the left of the robot would result in the left sensor average to be lower than the right sensor average, thus indicating a presence in the left half. The robot then moves away, i.e. to the right. The magnitude of turn can be computed as a function of the ratio of the left average and the right average.

$$a_r = \frac{\sum_{i=1}^{N} o_{ir}}{N}; \quad a_l = \frac{\sum_{i=1}^{N} o_{il}}{N}; \quad \phi_0 = f(\frac{a_r}{a_l}) \tag{4.1}$$

Here, $\phi_0$ refers to the heading variable of the snake, $a_l$ denotes the left sensor average and $a_r$ indicates the right sensor average. $o_{ir}$ and $o_{il}$ are the $i^{th}$ sensor outputs of the right and left arrays respectively. The function f is not fixed as the parameters can vary for different robots. It is best to discover the function f by experimentation on the actual robot.

To summarize, armed with the sensor representation of the world, the algorithm roughly estimates the direction of an approaching obstacle by considering the sensor outputs directionally and makes a turn towards the opposite direction depending on how close the obstacles is. Although we discuss the algorithm with just two directional divisions, left and right, more complex versions of this algorithm can be designed by expanding these directional divisions to four.

## 4.2 Automated Path Planning

Availability of meaningful information places bounds on how intelligent a system can get. That is equivalent to saying more intelligent systems require more rigorous world information to make the right decisions. This section investigates how better algorithms can be designed if presented apriori with the complete layout of the obstacle field.

### 4.2.1 Simulating Unknown Environments

Designing and testing generalized algorithms requires that we have a diverse set of scenarios to experiment with. But, that is infeasible in the real world, hence we resort to the virtual world. Computer simulations can help recreate the world in ways that would be impractical in reality. To this end, we devised a system of generating random obstacle fields which mirror real-world scenarios.

The snake-robot we are considering is restricted to the $2D$ plane and hence, perceives the world as a two dimensional entity. For our purpose, most real-world obstacles can be closely represented as either a cuboid or a cylinder. Cylinders and spheres appear the same to the robot as it can not perceive the curvature in the third dimension. We created a four-way randomizer to choose the number, type, size and location of obstacles which are $3D$ rendered in Gazebo to create truly unpredictable environments.

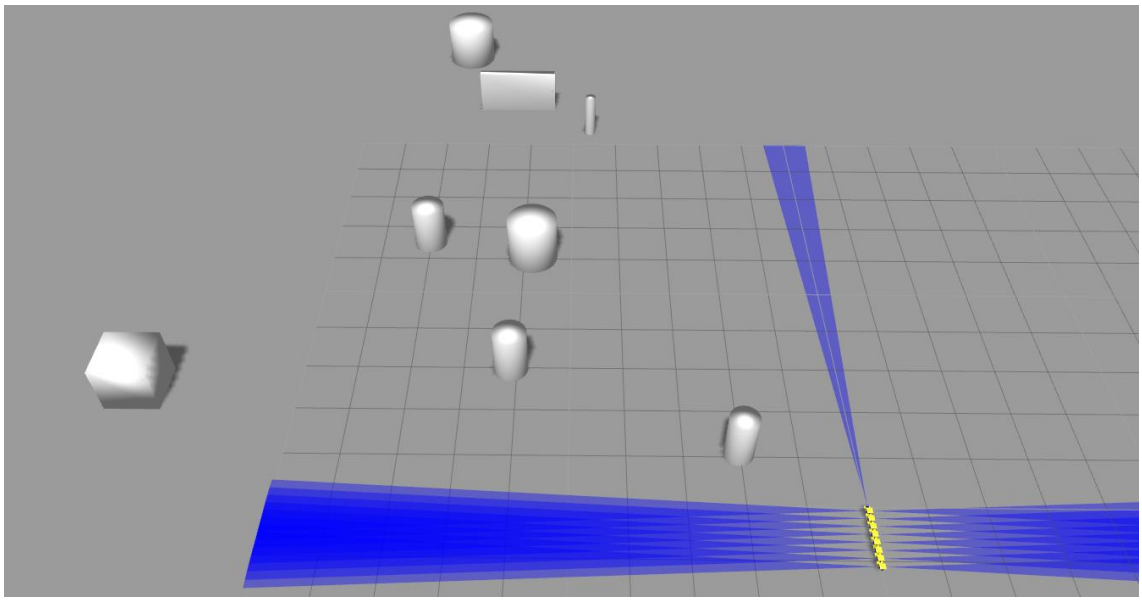

Figure 4.2: 3D rendered random obstacles

Gazebo rendering is not a straightforward task. Gazebo world files use XML which does not support complex functions like randomization. We got around this complication by writing Python scripts which, in turn, would write the code for Gazebo world files. For convenience, we set the starting point of the robot at the origin, i.e. $(0, 0)$, the destination at $(15, 15)$ and the obstacle field in between. This convention has been adhered to throughout the project.

### 4.2.2 Path-Planning Algorithms

It is known that the robot is restricted to the $2D$ plane and the obstacle map is also known beforehand. This implies that our path-planning algorithms can be designed to process top-view images of the obstacle field and draw out a path from start to destination while maneuvering around the obstacles in the path. For this project, we implemented two algorithms by leveraging well-known image processing techniques. The following sections discuss in detail the path-planning algorithms developed.

#### 4.2.2.1 Voronoi Graph Method

Voronoi tesselation is a mathematical method to partition a plane based on a distance metric from a set of points. What makes this technique our choice is a very useful property, which is, each Voronoi partition line bisects the distance between a pair of points. If these points represent obstacles, Voronoi partition lines give us the best paths between any two obstacles.

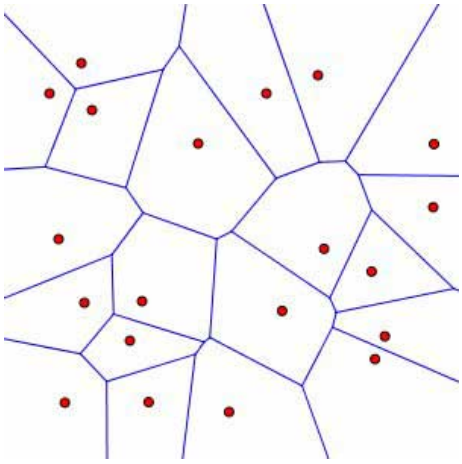

Figure 4.3: Voronoi tesselation

Applying Voronoi tesselation upon an image of 2D obstacles (see Note below) generates a network of Voronoi lines and Voronoi vertices, similar to the above image. Reinterpreting this network as a graph with Voronoi vertices as nodes and Voronoi lines as edges, we can employ shortest path algorithms to solve this graph and find out the best path from a start node (nearest to start point) to the finish node (nearest to destination). The edge cost can be defined as per the requirement. We define edge costs as a weighted combination of edge length and distance

between corresponding pair of obstacle site points. This ensures the algorithm chooses the shortest path while also considering the collision safety of the path.

The resulting algorithm chalks out a path like this:

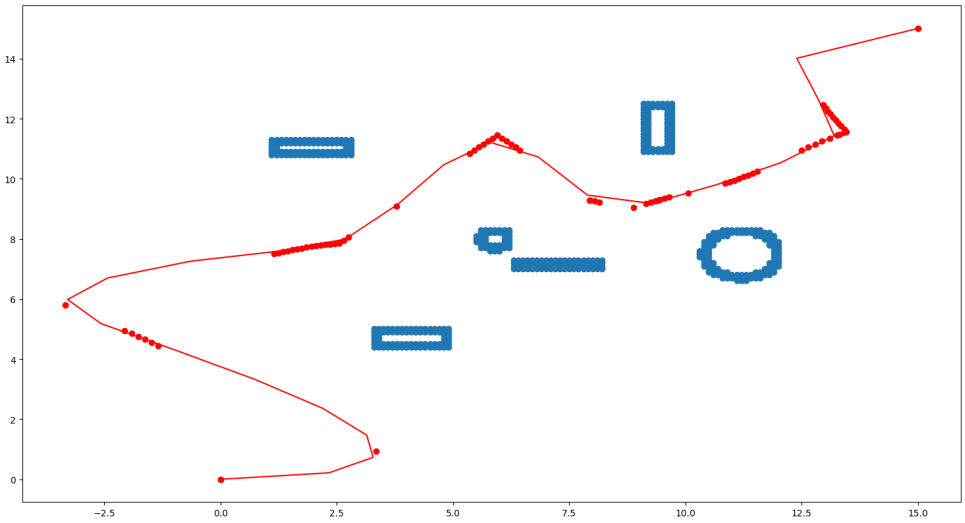


Figure 4.4: Voronoi Graph Method

**Note:** Two dimensional obstacles should not be represented by one dimensional points because it may lead to paths going through the obstacle. We use a idea we attribute to Paul Blaer (Columbia University) which is to represent the obstacle as a set of discrete boundary points, apply Voronoi tesselation and then eliminate paths within the boundary of an obstacle. Although we decided not to implement the last step in our algorithm, doing so would improve the effectiveness and we would suggest all further work to be done with it.

#### 4.2.2.2 Distance Transform Method

Distance Transform is an image-processing operator which processes binary images and returns pixel-wise distance from the nearest background pixel. It is commonly employed for applications like skeletonization. Distance transform, too, has a similar property to Voronoi diagrams, which is, it identifies the safest regions between obstacles.

If we visualize the obstacles as dark (background) regions in a binary image, distance transform highlights the paths with maximum physical distance from the obstacles. In addition, we employ another distance transform image, this time, with just the starting point as the dark (background) region. This ensures the algorithm is driven away from the starting point and towards the destination which is diagonally opposite. The gradient of the combined distance transforms will direct us towards the destination along the optimal paths. As a result, we achieve obstacle avoidance behavior

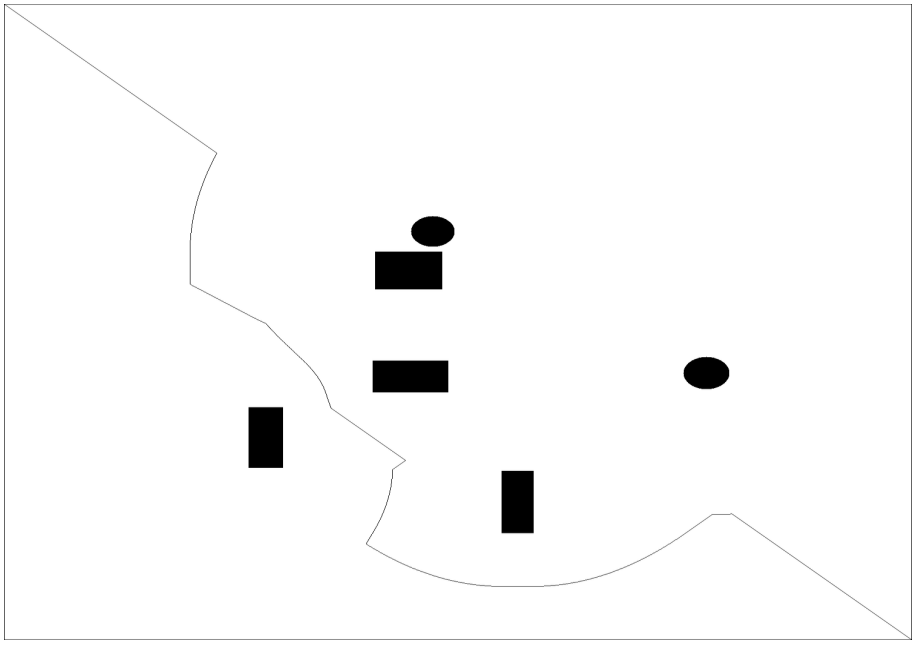

Figure 4.5: Distance Transform Method

It has been observed that, although the proposed algorithm works successfully most of the time, some configurations give rise to pockets of local minima which can cause gradient-based methods to get stuck. Any further work should experiment with momentum techniques to overcome this shortcoming.

### 4.2.3 Following Paths in Simulation

Although the automated path planning methods discussed above form the bulk of the work done in this domain, getting the simulated robot to actually follow the planned path is a small challenge of its own. We now discuss the method adopted for transferring the planned paths into ROS/Gazebo and controlling the robot to reliably follow them.

While discovering optimal paths by above algorithms, we periodically save the coordinates of points along the planned paths, called checkpoints. These checkpoints are saved into a file, from where they can be read by ROS controllers which operate the snake robot. The path following problem now reduces to getting from one checkpoint to the next. The angle needed to turn towards the next checkpoint is calculated by subtracting the slope between current checkpoint and the next checkpoint from the slope of the current heading direction. This way, we ensure the robot follows the planned path and avoids obstacles.

$$\begin{aligned} \phi_0 &= currentheading - (slope(currentlocation - nextcheckpoint)); \\ currentheading &= (slope(currentlocation - nextcheckpoint)) \end{aligned} \tag{4.2}$$

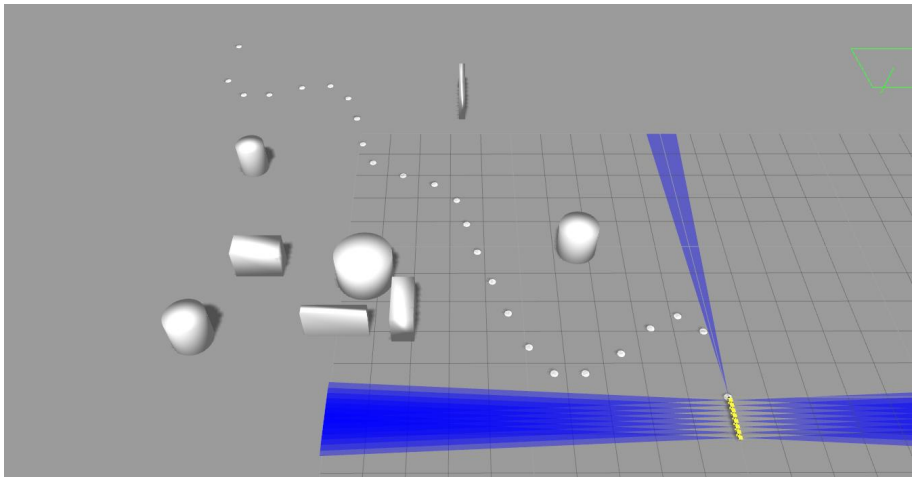

Figure 4.6: Simulation of Path Following

## 4.3 LSTM based Avoidance

The previous sections discussed methods to avoid obstacles blindly in the absence of overall information and introduced intelligent algorithms for path planning in the presence of global information. But, most practical scenarios lack such information. This chapter discusses an attempt to introduce intelligence into obstacle avoidance in spite of a lack of information using LSTMs.

We put LSTMs to the task of learning to dodge obstacles solely based on present and previous sensor data.

### 4.3.1 Data generation

Like any machine learning technique, LSTMs require data to learn the behavior we desire from it. In fact, owing to the time dependency, LSTMs require more data than conventional artificial neural networks. Unfortunately, there exists no standard reliable database of obstacle avoidance behavior available, hence we must generate our own training data. Fortunately, our automated path planning algorithms can be employed to run simulations of the robot avoiding obstacles and collect the necessary data for training the behavior.

Generation of training data requires careful planning of the details because any machine learning technique is only as good as its training data. The LSTM is meant to take in the 16-D sensor data and learn to predict the heading the robot is supposed to take in such a situation. Hence, the data generation would include recording a 16-D vector (a collection of 16 sensor outputs) and the heading of the robot at specific instants of time. We set up the data generation system such that it would create and simulate random scenarios, automatically plan and follow paths generated by Voronoi Graph Method and Distance Transform Method and record data at every change in heading direction (i.e. at every checkpoint). We also record data for every second in case of no direction change so that the LSTM can learn when to not act too.

Gazebo simulations are CPU intensive and take time to run. Simulations can be accelerated by setting the 'real time factor' to '0' and by setting 'gui' to 'false'. Over the course of 72 hours, we performed $1003$ simulations. After pruning the collected data to remove bad simulations, we finally had $73,589$ data samples over $962$ simulations for training, an average of $76.5$ data samples per simulation.

### 4.3.2 Training

We chose Keras as our platform for implementing the artificial neural networks, mainly due to its ease of use and online support. Various LSTM architecture were experimented with

before finally deciding on an architecture with three LSTM layers, containing 5, 10 and 50 nodes respectively, followed by three dense layers, with 50, 10 and 1 nodes respectively. The look-back (the number of previous time steps considered for each data point) was set at 5. The 16,000-parameter-network was trained for 200 epochs with a 0.15 validation split. Of the optimizers tested, RMSprop seemed to work the best for recurrent network training. All training was done on the IIT Guwahati supercomputer, Param Ishan.

We also document some of the problems faced during this stage. The heading variable ($\phi_0$), by nature, holds values of very low magnitude and can go negative too. This eliminates the 'sigmoidal', 'softplus' and 'ReLU' activation functions. Only 'linear' and 'tanh' functions can be set for activation. Due to low magnitude values, the validation loss has less scope of significantly decreasing. If time permits, we would like to investigate methods to overcome these limitations.

### 4.3.3 Testing & Results

Preliminary tests seem to indicate reasonably good correspondence between the predicted heading values by the trained LSTM model and the actual heading values. Upon testing over a 1000 test cases, the predicted heading was found to be off from the actual heading by a deviation equivalent to a physical 10 degrees. More rigorous training and testing could result in better prediction results and we hope to be able to do that by the end.

# Chapter 5

# Conclusion

Robotics is an exciting field in this day and age, and hyper-redundant robots, even more so. Along with tremendous potential for applications, these class of robots also have large need for further work. In this project, we explored one such aspect, namely, obstacle avoidance in snake robots. We created simulations for better understanding and easy prototyping of the complicated control algorithms necessary for these robots. We also designed avoidance algorithms with varying levels of intelligence depending on the world information accessible.

The rich potential this class of robots offers will hopefully draw more people towards this research topic and we hope they find our work on snake robots helpful for further research. More rigorous investigation of neural networks in navigation of snake robots is needed and can possibly change the way we work with robots. With this work, we hope to have inspired people to take up robotics research. We thank you, the reader for your attention.

# Bibliography


[1] S. Hirose and M. Mori, "Biologically inspired snake-like robots," in *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pp. 1–7, IEEE, 2004.

[2] G. S. Chirikjian, *Theory and applications of hyper-redundant robotic manipulators*. PhD thesis, California Institute of Technology, 1992.

[3] G. S. Chirikjian and J. W. Burdick, "The kinematics of hyper-redundant robot locomotion," *IEEE transactions on robotics and automation*, vol. 11, no. 6, pp. 781–793, 1995.

[4] J. P. Ostrowski, *The mechanics and control of undulatory robotic locomotion*. PhD thesis, California Institute of Technology, 1996.

[5] P. Prautsch and T. Mita, "Control and analysis of the gait of snake robots," in *Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on*, vol. 1, pp. 502–507, IEEE, 1999.

[6] P. Liljebäck, K. Y. Pettersen, O. Stavdahl, and J. T. Gravdahl, *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012.

[7] M. Yim, *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, stanford university, 1994.

[8] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, 2009.

[9] X. He, V. van Geirt, P. Gehlbach, R. Taylor, and I. Iordachita, "Iris: Integrated robotic intraocular snake," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1764–1769, IEEE, 2015.

[10] P. A. Vela, K. A. Morgansen, and J. W. Burdick, "Underwater locomotion from oscillatory shape deformations," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2, pp. 2074–2080, IEEE, 2002.

[11] K. A. Morgansen, B. I. Triplett, and D. J. Klein, "Geometric methods for modeling and control of free-swimming fin-actuated underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1184–1199, 2007.

[12] K. A. McIsaac and J. P. Ostrowski, "Motion planning for anguilliform locomotion," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 637–652, 2003.