
**Main Coursework**

Student: Arpit Sharma

P Number: P2613237

Lecturer: Uzor C

Module: Mobile Robots

Course: Intelligent Systems and Robotics

Date Due: 11 January 2021

**ABSTRACT**

The report presents an approach to completing the tasks of finding the middle of the room, finding the beacon, located Somewhere in the environment, going back to the middle of the room, and mapping the environment as it moves at the same time. A Pioneer 3-DX robot consisting of 16 inbuilt ultrasonic sensors and an external laser sensor, Hokuyo (fast) laser sensor mounted on its top and facing the front of the robot is simulated with freely steering the virtual environment and not touching or bashing into the wall. The solution pulls the extensive functionality of the CoppeliaSim (previously known as V-REP) virtual robot experimentation platform [1] to provide the simulated physical environment layer, including wall and mobile robot objects. The programming has been done in Lua and edited in Visual Studio, which manages environment perception and navigation. A final assessment is made, by trying different starting positions of the robot for task 1, changing the position of the beacon, changing the structure of the environment by adding objects, making it difficult for the robot to find the beacon in task 3, and finding the room's center in task 4 and noticing the responses of the robot. It is realized that the VFH* algorithm is far superior to VFH+ and VFH as it gives more value to future decisions than the immediate decisions. Recommendations include more refined laser sensor calibration and further research into VFH (Virtual Control Histogram).

## INTRODUCTION

The goal of this study is to develop and simulate a Pioneer P3dx robot in the CoppeliaSim simulator to perform the following tasks:

1. The robot starts from any position in the room, move, and by using the sensors, find the center of the room.
2. Then, it aligns itself with the exit of the room and comes out in a straight line without bashing into the doorway.
3. Then, the robot starts exploring the environment to find the beacon and avoids objects on its way. Once it finds the beacon, the robot stops near the beacon at fewer than 0.5 meters.
4. Once the beacon is found, the robot sets off to find the middle of the room without bumping to the sides and stops exactly in the middle of the room
5. During these tasks, the robot maps the environment using data from the sensors.

The ambiguities in the environment are the starting positions of the robot, the beacon’s position, and the number and types of obstacles. This paper is organized as follows. Section 1 details the setup required for the experiment, while section 2 provides details on the robot model used. Sections 3 covers task 1. Section 4 provides some technical detail on the task 2 implementation. Section 5 presents task 3, Section 6 consists of task 4 and task 5, Section 7 is conclusion, and section 8 closes with the final evaluation and thoughts.

# SECTION 1

## Setup

The basic setup necessary for running the simulation requires to copy and paste the plugin file (plugin is used in task 4 for implementing VFH* algorithm) provided in the folder named ‘plugin’ in the code’s zip file to the Coppeliasim’s installation directory as such:

- simExtVFHp.dll, if working on Windows
- libsimExtVFHp.so, if working on Linux
- libsimExtVFHp.dylib, if working on Mac

The IDE provided by the Coppelaisim is not very user-friendly as there were no formatting options, autocompletion tools, and git GUI. As everyone has a favourite editor, the following steps can let anyone use that with Coppeliasim.

1. Create a file with a .lua extension in the workspace.
2. Copy and paste the entire code from the script of the robot in Coppeliasim to the created file.
3. In the script, delete everything, and paste the following code:

   *local f = assert(load file("Full path of the created file"))*

   *return f()*

4. Close the script file.
5. Now, open the Lua file in any editor of choice and can start editing.

# SECTION 1

## Simulated Environment

Physical and geometric characteristics of the simulated world, including ‘collidable’, ‘static’, and ‘responsive’ walls, beacon and the obstacles, and the scene (Fig. 1) is modelled in CoppeliaSim. The environment is 15m by15m with varying obstacles and 80cm high walls that make up the boundaries of the environment and the inner room. The floor is flat and smooth. No constraints like friction, air pressure, power consumption is considered in the experiment.

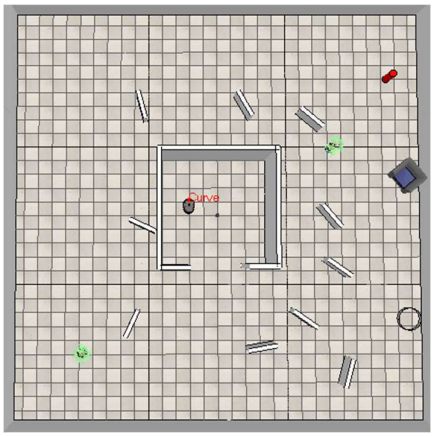



*Fig.1 CoppeliaSim scene layout*

# SECTION 2

## The Robot Model

Modelled in CoppeliaSim and used here is the well-known Pioneer P3DX robotics research platform [2], with the Hokuyo (fast) URG 04LX UG01 laser sensor mounted on its top as shown in figure 4. Pioneer P3DX robot (Figure 2) with 2 rear wheels, powered independently, and an omnidirectional caster wheel in the front, is modelled in the virtual environment provided by CoppeliaSim.The dimensions of the robot, shown in figure 6, are important for the calculations of the distance from the obstacles, beacon in task 3, and for selecting the right direction to move in task 4.

**Building the Model**

Hokuyo (fast) laser sensor found in the sensor category of the Coppeliasim's model browser is used. There are 3 models available in the list, the second has been selected, and the sensor has been made the child of the robot in the scene hierarchy.

*Fig.2 A physical Pioneer P3DX robot*

*Fig.3 a physical Hokuyo URG 04LX UG01*

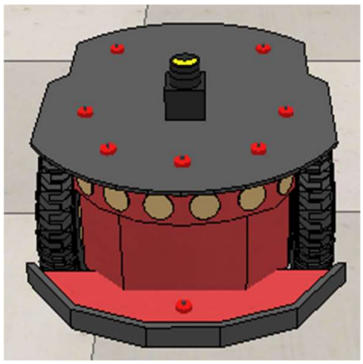


*Fig.4 A virtual Pioneer P3DX robot with the laser sensor mounted on its top*

*Fig.5 A virtual Hokuyo URG 04LX UG01*

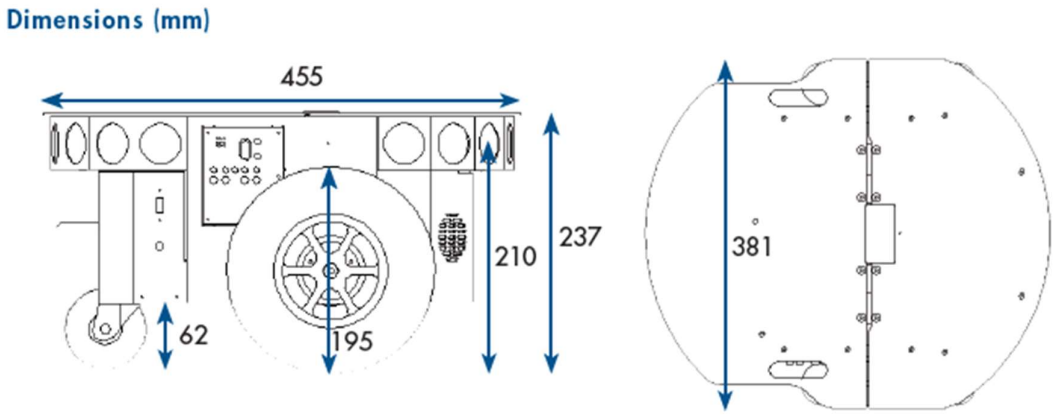



*Fig.6 Dimensions of Pioneer P3DX robot*

**Actuators**

The robot consists of 3 wheels in a triangle. Two larger and rear wheels are powered independently with bidirectional geared DC motors for getting the required torque, and the passive caster wheel in the front is omnidirectional. The combination makes the robot capable of rotating about its axis, moving and turning in any direction. In Coppeliasim, the motors are mimicked by the revolute joints which require input in radians (positive value for the forward direction, a negative value for reverse, and 0 for stopping).

**Sensors**

The robot consists of 16 inbuilt ultrasonic sensors, numbered as shown in figure 7. The sensors work by emitting sound waves and detecting the reflection received to calculate the distance from the reflecting body which enables the robot to detect the distance from any object around it in any direction because of sensors surrounding the robot. By default, these sensors are a cone type with max range 1m that “allows for the best and most precise modelling of most proximity sensors” [3]. When the robot is in contact with the wall, the sensor, perpendicular to the wall and facing it, reads 0.09(not 0) because of its offset position. So, the structure of the robot is important for calculating the threshold distance.

Other than this, an external laser sensor, Hokuyo URG 04LX UG01 has been added to the robot which is used for task 4 and task 5.

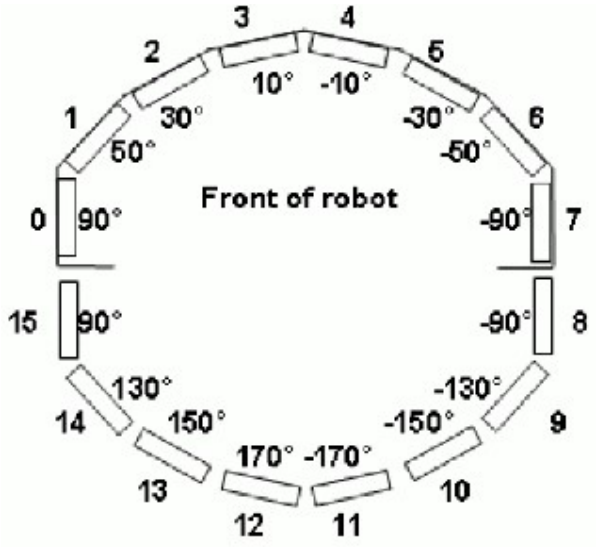



*Fig.7 Numbering of ultrasonic sensors on the robot with angles*

## SECTION 3

### Task 1

Task one was to find the center of the room starting from a random position inside the room. So, the task was divided into 2 subtasks:

1. Measuring the length and breadth of the room
2. Finding the center of the room

**Measuring the Length And Breadth Of The Room**
So, the robot starts from a random position inside the room and the robot performs the following steps in the proper sequence:

1. The robot goes straight to the wall in its front and then turns at 90° (like shown in 'position a' in figure 8).
2. it moves straight following the wall till there is another wall in front (i.e., it is a corner), it turns 90° (like shown in 'position b' in figure 8) and notes the current position of the robot (first co-ordinates).
3. Then, it moves straight (following the wall) till another wall in front of it is detected again. And when that happens (as shown in 'position c' in figure 8), it notes it is the current position (second co-ordinates). So, the distance between the first co-ordinates and the second co-ordinates is the length or breadth of the room minus twice the threshold distance of the robot (measured length).
4. Again, it moves straight till there is a wall in front (like shown in 'position d' in figure 8) and notes its current position (third co-ordinates). So, the distance between the second co-ordinates and the third co-ordinates must be measured breadth if the last

measurement was the measured length, and measured length if the last one was measured breadth.

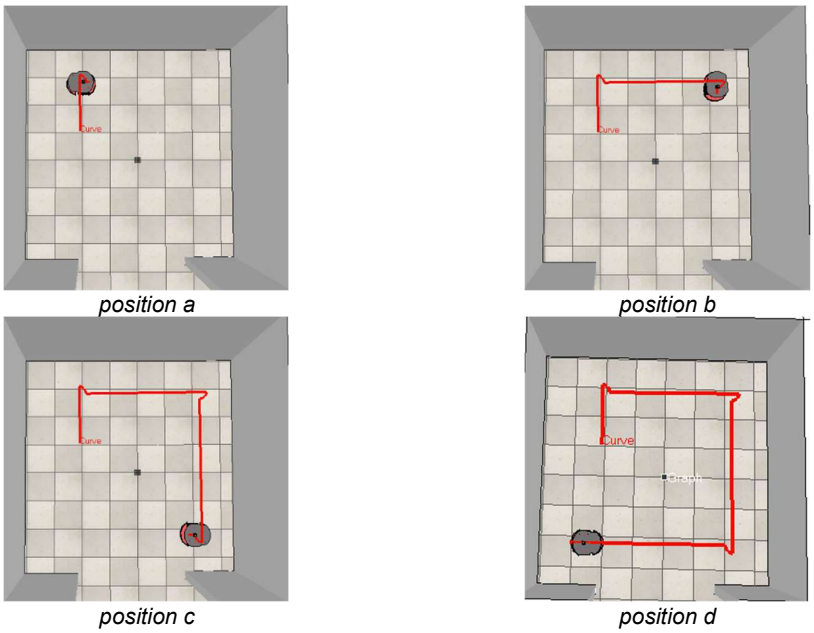



*Fig.8 Measuring Walls*

**Finding the Center Of The Room**

After measuring the wall is done, we have the measured length and measured breadth of the room. Although measured length can be the measured breadth and vice-versa, this detail is not required to know. The measured length and measured breadth are used for calculating the center of the room according to the following logic:

$$measured\ length = Actual\ Length - 2 * (threshold\ distance)$$

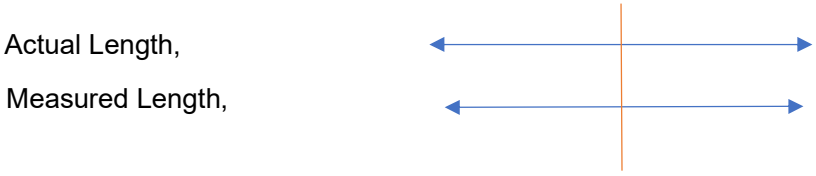
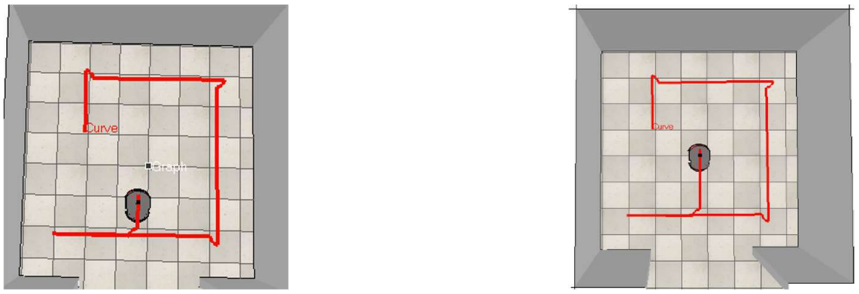



*Fig.9 Measuring Walls*

The middle of the two lengths is still the same and is indeed the x or y coordinate of the center of the room (as shown in figure 9). Similarly, it is true for the measured breadth. That is why measured length and measured breadth can be used for finding the center of the room.

So, to reach the center of the room, the robot performs the following steps in the proper sequence:

1. the robot moves in the opposite direction for a distance equal to half of the second measurement and turns 90°. (like shown in ‘position a’ in figure 10)
2. moves straight for a distance equal to half of the first measurement (like shown in ‘position b’ in figure 10).
3. Hence The center of the room is found. It notes its current position co-ordinates which will be used in task 4 for finding its way back to the center of the room.




*position a* *position b*

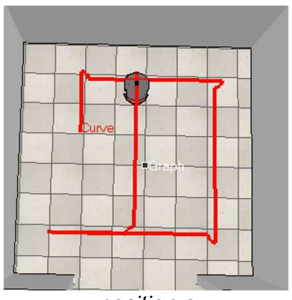
*Fig.10 Measuring Walls*

## SECTION 4

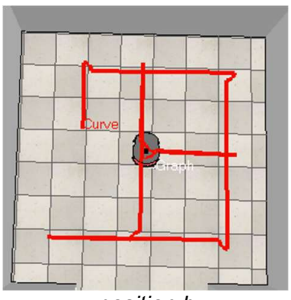
### Task 2

Task 2 was for the robot to align itself with the exit and come out of the room in a straight line without bashing into the doorway. The robot was programmed to perform the following steps in the Sequence to complete the task.

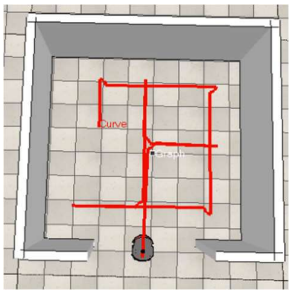
1. D (distance to move in a straight line once), is made equal to half the first measurement
2. From the center of the room, the robot moves in a straight line for a distance D (as shown in ‘position a’ in figure 11).
3. If it detects a wall in front of it, it moves the same distance back to reach the center of the room again and turns 90° (as shown in ‘position b’ in figure 11), otherwise, if no wall is detected, it means, it is the doorway, So, it comes out of the room in a straight line (as shown in ‘position c’ in figure 11) and the rest of the steps are skipped as the task is over.
4. Distance, D is toggled. if it is equal to half of the first measurement, it is made equal to the second measurement and vice – versa.
5. Again, steps are performed in sequence from step 2.




*position a*




*position b*




*position c*

*Fig.11 Measuring Walls*

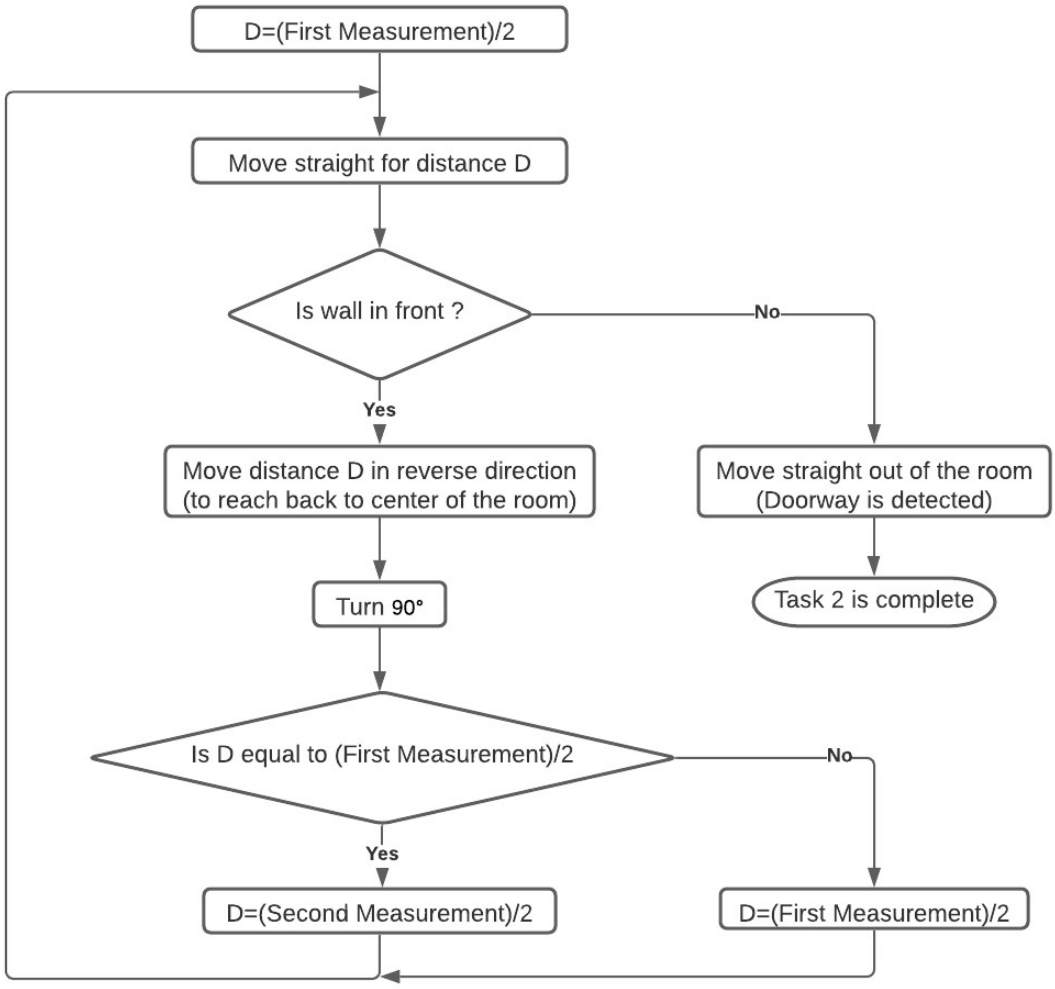



*Fig.12 Flowchart for Task 2*

## SECTION 5

### Task 3

Task 3 was to find the beacon located somewhere in the environment outside the room. Then, stop near the beacon (less than 0.5 meters from it) So, for completing this task, the PID controller [4] designed in 'Lab 6 Portfolio' was used for following a wall, curved or angular surfaces as it is much more efficient and stable than the wall follower designed in 'Lab 4 Portfolio' which used bang-bang approach.

Beacon is a measurable detectable collidable, and renderable object in the scene as shown in figure 13.

The robot after exiting the room in task 2 finds a wall, and aligns itself with it, and starts following the wall on its right.

Now, it is constantly checking for the objects on its left by the readings of the ultrasonic sensors on its left. First, second, third, and sixteenth ultrasonic sensors of the robot have been deployed for this job (numbering is according to figure 7). So, when an object on its left is detected (beacon), it turns towards it, till ultrasonic sensors in front of the robot read positive (i.e., it faces the beacon), and then, moves towards it till the distance from it is greater than 0.5 meters from the beacon and then stops. Hence, task 3 is complete.

A threshold of 0.4 was chosen to maintain 0.4 meters from the wall on its right. Obstacles (sofa and ring) were put on the way to the beacon. The obstacles were simply avoided by the robot with the PID controller alone. The proportional gain (it makes the robot turn proportionally to its distance from the desired trajectory i.e. cross-track error) of 397, the integral gain ("it acts as a basket in which the loop stores all measured errors" [5]) of 53, and the derivative gain ("It acts as a brake or dampener on the control effort" [6]) of 55 were used to tune the controller.

The only limitation of the approach is that the beacon must be placed close to the wall. if the beacon is situated somewhere in the middle of the scene away from boundary walls, the robot would not be able to find it. Knowing the beacon's position coordinates from the starting would have made it much simpler and the approach used in task 4 could be used but, we were not supposed to know the beacon's position from the start, so, this approach had to be used.

Another approach possible could be to make the robot wander randomly and map the environment, and upon detecting the beacon, stop near it. But, again, it had to know the shape of the beacon from the starting.

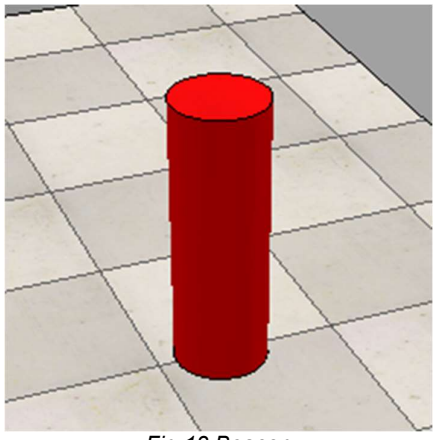


*Fig.13 Beacon*

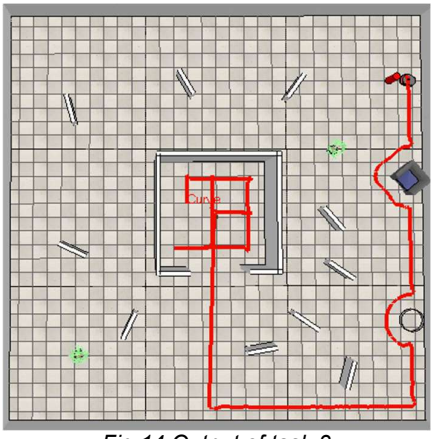


*Fig.14 Output of task 3*

## SECTION 6

### Task 4 and Task 5

Now, at this stage, the beacon has been found, and the position coordinates of the middle of the room (calculated in task 2) are known. So, again, go to where the robot started, the canter of the room, these Coordinates are used as a goal for the Vector Field Histogram (VFH*) algorithm.

So, for understanding VFH* algorithm, VFH algorithm, and VFH+ algorithm is required to be understood as the VFH* is just an enhanced type of VFH+, and so is the VFH+ of VFH.

**Vector Field Histogram (VFH)**

It uses a 2D occupancy map or cell histogram (shown in figure 15) that allows incorporating "unreliable" measurements from the laser sensor. It is based on the VFF (Virtual Field Force) technique. Then, the cell histogram is reduced to a polar histogram. The sectors of the histogram that do not exist (their values are below the threshold values) are considered as free sectors. Therefore, the threshold is very critical and must be properly tuned.

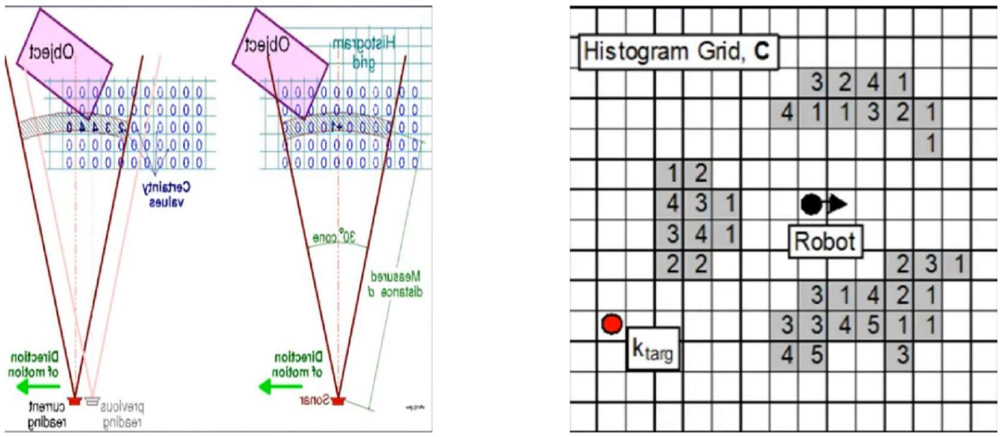



*Fig.15 Cell Histogram*

**Candidate Valley Detection:** Then, it detects valleys (free sectors) (valley size depends on the environment of the robot. In the polar histogram (shown in figure 16), to propose candidate directions [7] to move (lower values of the histogram in each direction indicate that such direction is less likely to point towards an obstacle). The direction of the goal (middle of the inner room) configuration is also considered. once valleys (set of sectors below the threshold) have been found, the one candidate direction that is closest to the target is selected and within that valley, a direction according to the valley size is selected depending on its wide or narrow valley (shown in figure 17). Robot size is also considered to avoid implementing a lowpass filter.

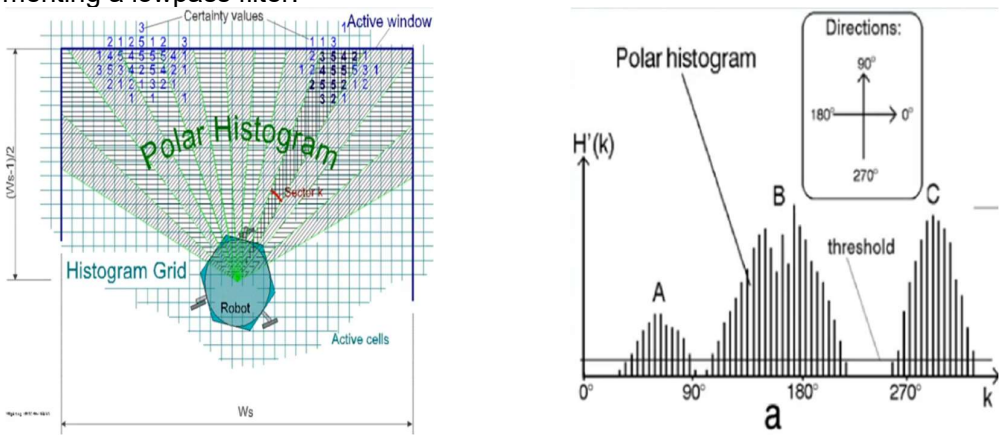



*Fig.16 Polar Histogram*

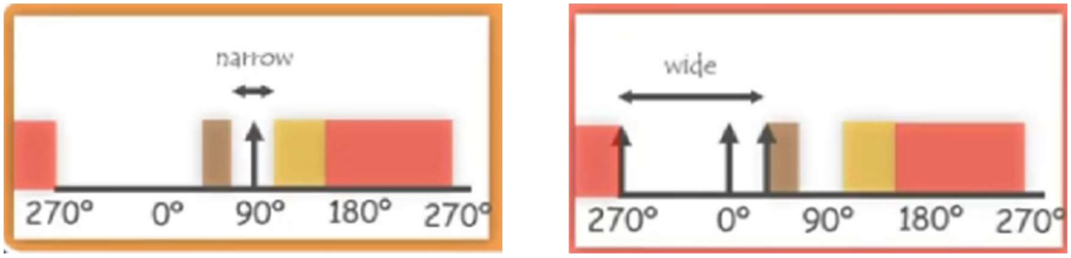



*Fig.17 Narrow Valley*

**Vector Field Histogram (VFH+)**
VFH+ algorithm [8] improves the VFH algorithm by considering:

**Binary Histogram**: Computes a binarized histogram (shown in figure 18) by using two thresholds with hysteresis that introduces smoothness in motion planning. Also, the robot size is considered, to avoid implementing low-pass filters.

It Uses two thresholds to determine if a sector is occupied or free. if the value lies between the threshold values, then the value of the previous iteration (hysteresis effect) is used. It transforms the accumulated values to 0s and 1s depending on whether they are above or below the threshold Between these thresholds' hysteresis is used so that the previous value or the values in the previous iteration is used.

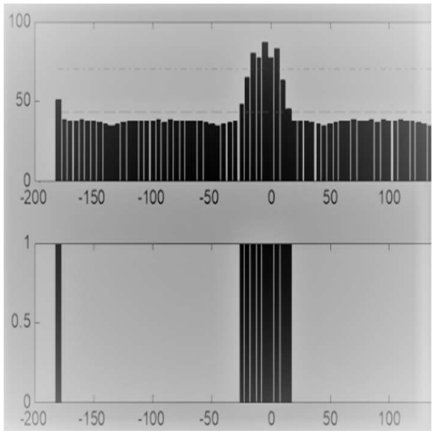
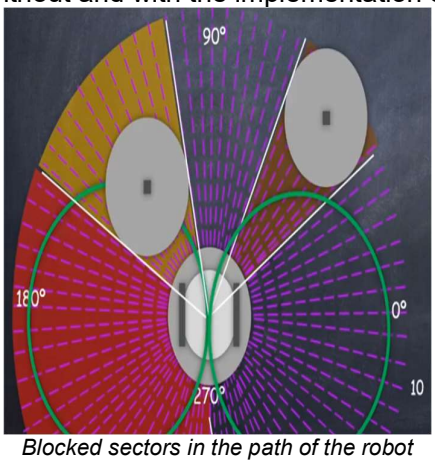
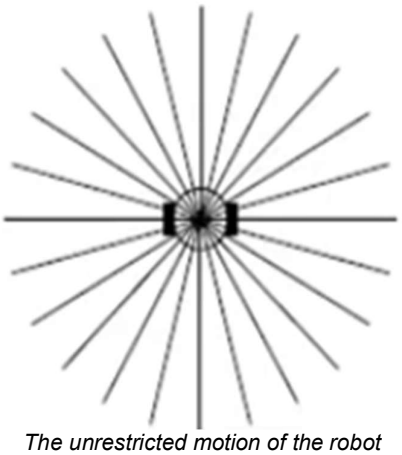



*Fig.18 Binary Histogram*

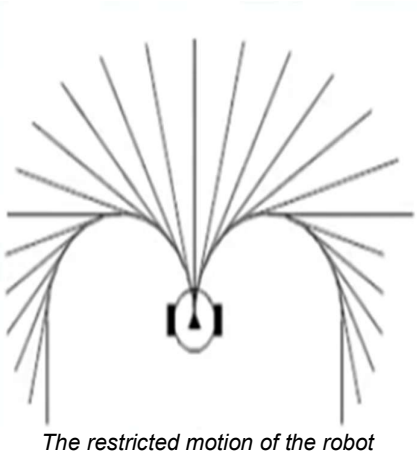
**Masked Histogram For VFH+:** Masked histogram uses the concept of unreachable directions considering that the robot has a limited turning radius due to the kinematic or dynamic constraints. It blocks those directions that the robot will not be able to reach without collision by considering kinematic and dynamic constraints. Figure 24 shows the motion of the robot without and with the implementation of a masked histogram.




*Blocked sectors in the path of the robot*



*The unrestricted motion of the robot*



*The restricted motion of the robot*

*Fig.24*

**Cost function**: The selection criterion combines candidate directions that allow defining behaviour. By modifying the cost function, as the behaviour changes. Cost function depends on target distance, the robot's orientation, and the previous direction distance. Candidate direction with the minimum cost function is selected.

**Vector Field Histogram (VFH*)**

The VFH* algorithm improves VFH+, and it is not a purely local algorithm anymore. VFH+ is "myopic" in certain circumstances since it cannot distinguish between two possible candidate addresses: one good and the other bad. VFH* includes the A* algorithm [9], to evaluate the next steps before making the decision. Candidate directions are used to predict the robot's position one step ahead. From that new position, new candidate directions are computed using the VHF+ algorithm. Repeating the previous step several times, a tree with candidate directions can be computed. The cost after "n" steps is used to select the best one-step candidate direction. So, for this technique, we need a cell histogram, whose values are updated on the range sensors measurements after several passes and they will provide, a certain of the cell being occupied or not. This process can be improved if we use occupancy maps which provide the likelihood of it being occupied. A two-dimensional cell histogram is converted into a polar (one-dimensional) histogram. In this process, the size of the robot is taken into consideration, which avoids having to use a low-pass filter to smooth the polar histogram. for each cell, magnitude is computed, depending on its distance from the robot and its occupancy likelihood, and then the cell magnitudes are accumulated based on their corresponding sector and the size of the sector is the fundamental parameter of the algorithm as well. Uses a tree of candidate directions (as shown in figure 25) to determine the best one-step decision to take. Hence, tree nodes are predicted positions of the robot based on a candidate direction. VFH* is introduced so that future decisions value more than the immediate ones.

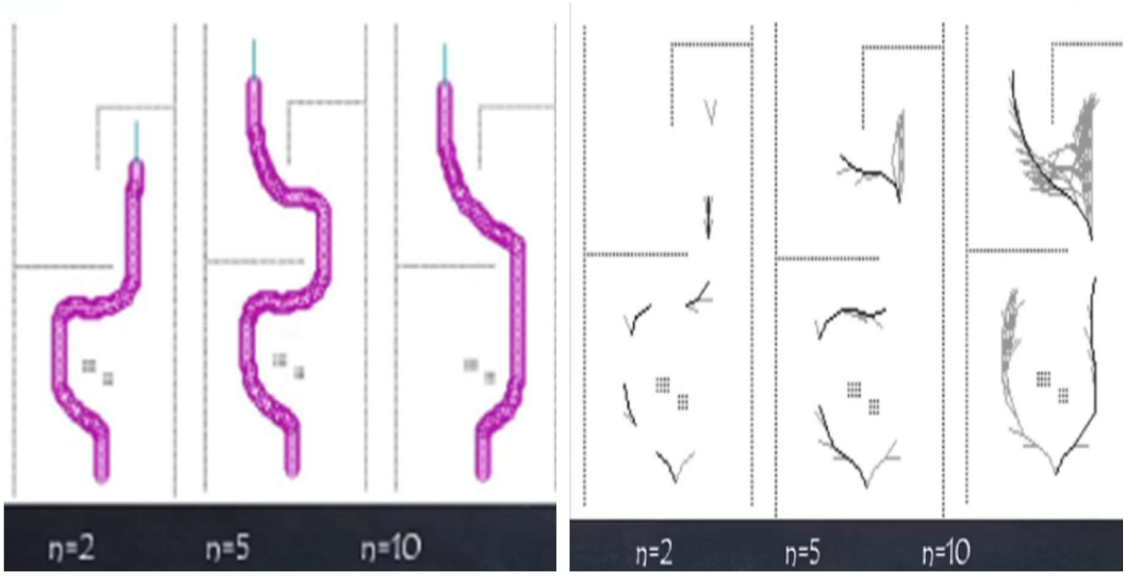



*Fig.25 Tree of candidate directions*

**Masked Polar Histogram For VFH***

Kinematic and dynamic constraints: Some sectors are blocked because they cannot be reached without collision. To reach a free Sector, decided according to the binary histogram, if we need to reach that sector, the robot must necessarily go through another sector that might be blocked then, in that case, those sectors will also be blocked. Because to reach these directions, the robot must necessarily pass through the orange region, and because of the Minimum radius, which is deleted in the green, it is not possible without colliding green circles are the minimum radius.

Once, we have the masked histogram, we can detect the valleys.

Narrow and wide valley distinction:

***Narrow Valley***: Only one candidate direction: one in the middle.

***Wide Valley***: Three possible directions: Left, Right, or straight towards the goal (if it is inside the valley).

So, implementing the algorithm for the robot for completing this task, the VFH* plugin by Leopoldo Armesto has been used. The candidate directions, target direction, and selected direction are shown in figure 26 as:

1. Green lines represent the mask histogram to search for directions that are free of obstacles.
2. We get many candidate directions (longlines).
3. Black is the line always pointing towards the target.
4. So, following black isn't possible. so, the only direction we take is blue (short direction).
5. We have 2 candidate directions, we are selecting the one that is closer to the target.
6. When we have the target in the field of view, there is a candidate direction pointing directly towards the target.
7. Valley size depends on the environment of the robot.

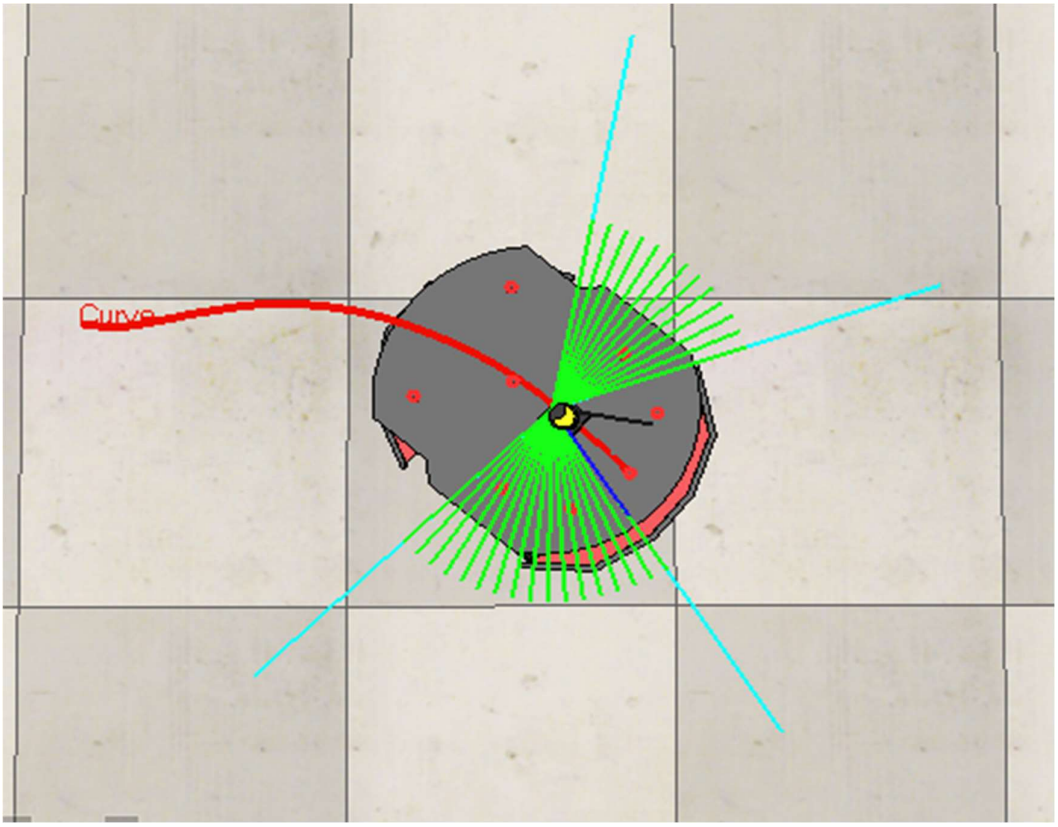



*Fig.26 Measuring Walls*

So, for finding the middle of the room (as shown in figure 27), the robot, by implementing the VFH* algorithm, avoids the obstacles on its way, and ultimately finds its goal (as shown in figure 28).

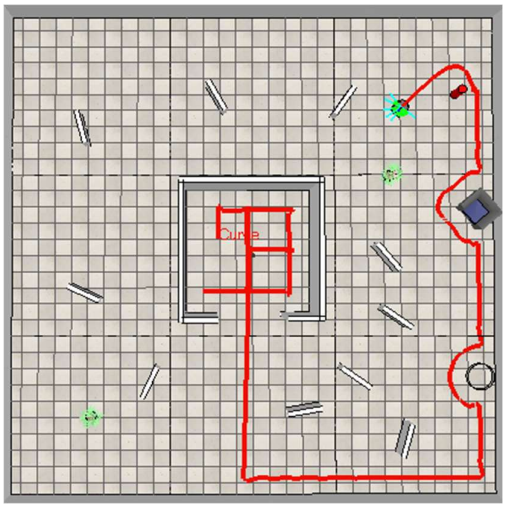


*Fig.27 The robot finding the middle of the room*

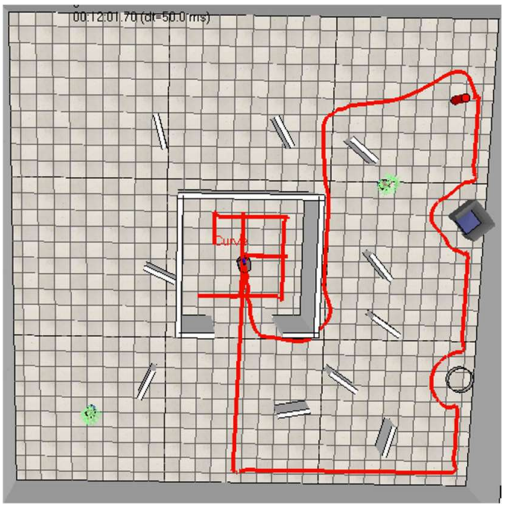


*Fig.28 The robot having completed task 4*

## SECTION 7

## Conclusions

In task 1, the robot calculates the coordinates of the center of the room by measuring the dimensions of the walls (length and breadth) and then finds the center by moving half the distance in the respective directions.

In task 2, for finding the doorway, the robot moves and checks for a wall in all four directions with the help of its front ultrasonic sensors, and where it finds it missing, it exits the room, hence, completing the task.

In task 3, for finding the beacon, the PID controller is used for moving along a wall and constantly checks for the beacon on its left with the help of its left ultrasonic sensors. On detecting the beacon, it goes towards it and stops near it at 0.5m using its front ultrasonic Sensors.

In task 4 and task 5, the VFH* algorithm works like a charm as it foresees its future positions unlike VFH and VFH+ and hence, does better path planning and is the reason, the length of the path becomes shorter and smooth, and mainly, the robot does not get stuck anywhere in the process.

## SECTION 8

## Final Thoughts

The issue I found is with the Coppiliasim is that, for using sensors based on vision sensors, like laser sensors, the simulation significantly slows down. To fasten the simulation, it must be performed on a system having a good graphic card if other parameters of the sensor like the number of points to scan, scan angle, etc cannot be changed. But, I hope Coppeliasim finds a workaround for slow processors in the future.

Overall, there was so much to learn in this experiment, especially from task 4 and task 5, and these Several weeks of experimentation finally led to the completion Of the tasks.

Once, having mastered these tasks, it is a piece of cake for implementing them on a physical Pioneer 3-DX robot or any other robot (after minor modifications).

## REFERENCES


[1] VREP, “V-REP Virtual Robot Experimentation Platform Home Page,” Available online: http:/'www.coppeliarobotics.com.

[2] P. P3DX, “Pioneer P3DX Rev A Data Sheet.” Adept Mobile Robots, Available online: https: www. Generationrobots.com/media/Pioneer3DX-P3DX-RevA. pdf.

[3] http:/www.coppeliarobotics.com, “Proximity sensor types and mode of operation,” Available online: https://www.coppeliarobotics.com/helpFiles/en/proximitySensors.htm

[4] https://www.ni.com, “PID Theory Explained,” Available online: https://www.ni.com/en-gb/innovations/white-papers/06/pid-theory-explained.html

[5] Kumar, Rakesh, “Introduction to PID control”, Available online: https://www.machinedesign.com/automation-iiot/sensors/article/21831887/introduction-to-pid-control

[6] Welender, Peter, “Understanding Derivative in PID Control”,Available online:https://www.controleng.com/articles/understanding-derivative-in-pid-control/

[7] Borenstein, J.; Koren, Y. (1991). "The vector field histogram-fast obstacle avoid mobile robots". IEEE Transactions on Robotics and Automation. 7 (3): 278-288. Available online: https://ieeexplore.ieee.org/document/88137

[8] Borenstein, J. (1998), “VFH+: reliable obstacle avoidance for fast mobile Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference Available online: https://ieeexplore.ieee.org/document/677362

[9] Ulrich, I.; Borenstein, J. (2000), “VFH: local obstacle avoidance with look-ahead verification” Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference