REPORT AUTONOMOUS NAVIGATION IN MAGNETIC CRAWLER

Objective: to develop a solution to enable autonomous navigation in the magnetic crawler.

**Autonomous navigation** is an attempt to provide a robotic system the ability to determine its location and be able to plan a path to reach its goal without needing human interference.

This could be broken down into the following steps -

**Act**

control the vehicle to follow the path

**Sense**

Collect data about the environments

**Plan**

plan a path to reach the goal

**Perceive**

Localize

Estimate state

Mapping

Track objects

**How to achieve autonomy:**

The degree to which a robotic system possesses autonomy is based on the use case could range from simple assistance during manual/teleoperation to complete autonomy.

|  |  |
| --- | --- |
| Heuristics | Optimization |
| * Follows a set of practical rules or behaviors. * Not Guaranteed to be optimal. * Does not need complete information about the environment. * Simple rule like keep the wall to the left to turn away when you sence obstacles. | * Requires more information on the environment. * The planning is achieved from the maximization or minimization of some value. * More effective but more computationally costly. |

We can understand the approaches with some examples – maze-solving vehicles and robotic vacuums.

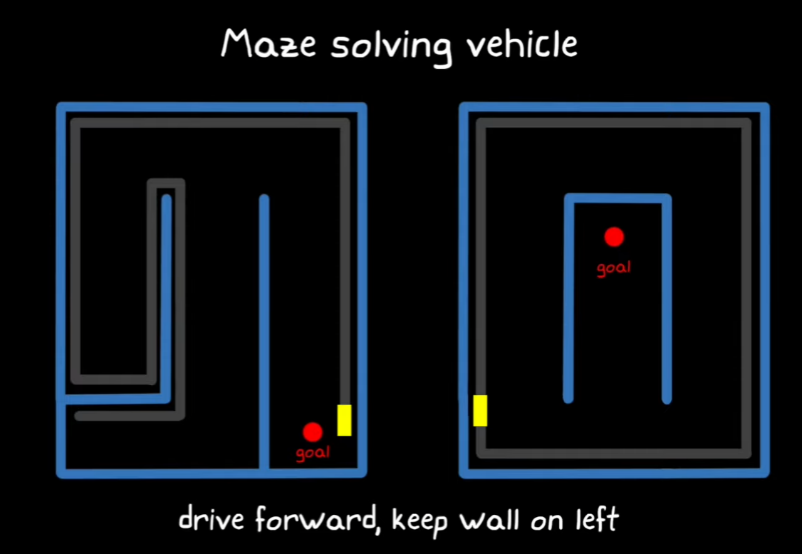
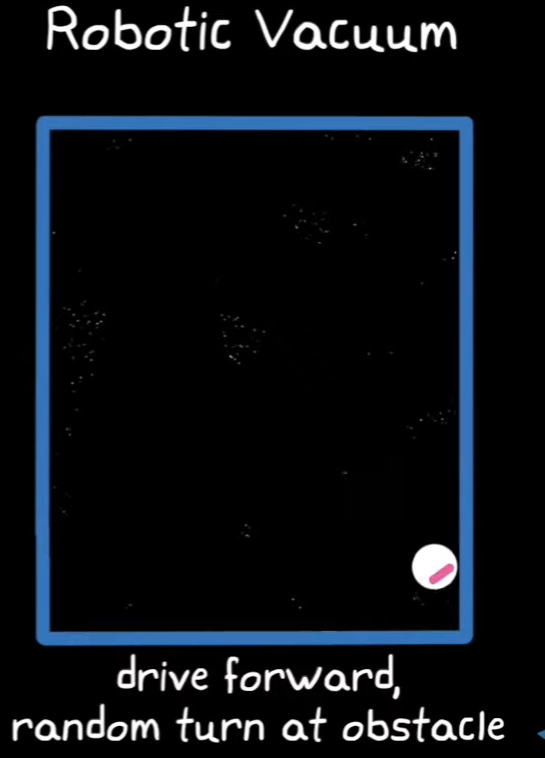
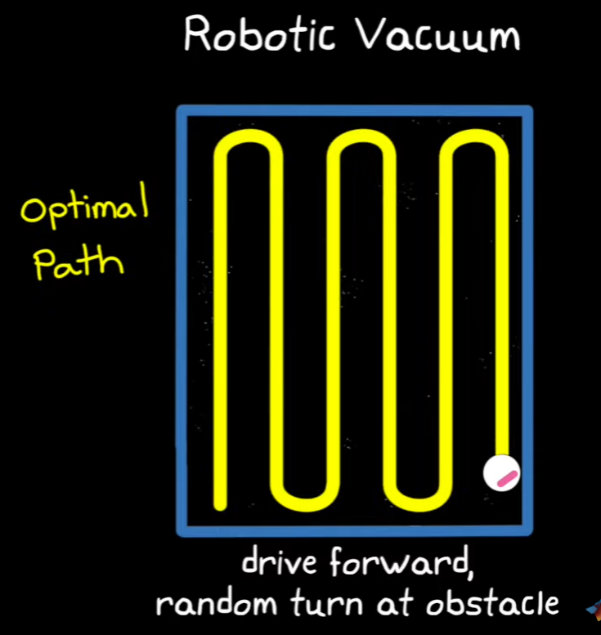


Fig: illustration of a maze-solving vehicle instructed to keep the wall to the left (heuristic-based approach).

Imagine you are building a robot to navigate through a maze, operated with the instruction that it always must ensure the wall has to be on the left it may achieve the objective of reaching the goal in a very limited number of use cases. The case (on the right) illustrates a case where this may fail and shows that this approach has a threat of the vehicle getting stuck in an endless loop.

Robotic vacuum – a robotic vacuum cleaner to clean the entire floor in the room could be achieved by adding a simple heuristic - keep driving forward but once encountered with an obstacle, turn away at a random angle and continue. As time progresses the chances of the robot having cleaned the entire floor reaches 100%. Although the robot did not take the optimal path the objective was accomplished.



**Optimized approaches**

This requires the robot to build or update the already available model of the environment in which it operates. Then it must plan the path that it would take to achieve the goal through some optimization algorithm. A very famous example of this the self-driving cars.

It is usually very difficult to build fully optimal systems at all times because, in the real world, it is impossible to model or estimate every little interaction the robot has with its environment. Therefore most modern systems combine the two methods where the two approaches complement each other.

**Types of sensors in robotic systems.**

Autonomous navigation systems rely on a variety of sensors to perceive and interact with their environment. Here are the main types of sensors typically employed:

1. **Vision Sensors**

* **Cameras**: Capture images or video, providing rich visual information. Used for object detection, recognition, and scene understanding.
* **Stereo Cameras**: Two or more cameras spaced apart to estimate depth by comparing images.
* **Depth Cameras**: Such as RGB-D cameras, provide color images along with depth information, useful for 3D mapping and obstacle detection.

2. **LIDAR (Light Detection and Ranging)**

Uses laser beams to measure distances to objects, creating detailed 3D maps of the environment. Essential for precise localization, obstacle detection, and mapping.

3. **Radar**

Uses radio waves to detect objects and measure their speed and distance. Effective in various weather conditions and for detecting objects at longer ranges.

4. **Ultrasonic Sensors**

Emit ultrasonic waves and measure the time it takes for the echoes to return. Commonly used for short-range obstacle detection and parking assistance.

5. **GPS (Global Positioning System)**

Provides global position information, essential for outdoor navigation. Often combined with other sensors for improved accuracy.

6. **IMU (Inertial Measurement Unit)**

Combines accelerometers and gyroscopes to measure acceleration and rotational rates. Provides information on the orientation and movement of the vehicle, aiding in dead-reckoning and stability control.

**7. Odometry Sensors**

Measure the distance traveled based on wheel rotations or other mechanical means. Useful for estimating position changes over time.

**8. Sonar**

Similar to ultrasonic sensors but generally used in underwater applications. Measures the distance to objects using sound waves.

**Sensor Fusion**: Combining data from multiple sensors to improve the overall perception and reliability of the navigation system. Techniques like Kalman filtering and particle filtering are often employed for sensor fusion.

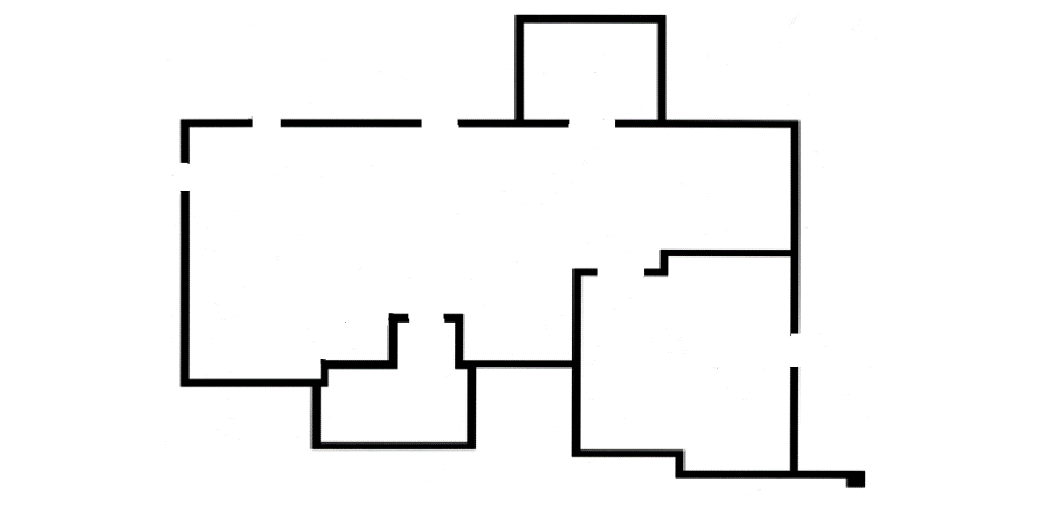
By integrating these sensors, an autonomous navigation system can achieve robust perception, accurate localization, and reliable obstacle detection, enabling safe and efficient navigation in diverse environments.

**SIMULATING LIDAR USING PYGAME –**

lidar uses light rays to approximate the distance of the nearest obstacle. The lidar uses the time the light ray takes to bounce off the obstacle and return to estimate its distance from the sensors.

The LIDAR is mounted on a rotating stage that allows us to look for obstacles in all directions. Therefore, LIDAR offers a reliable way to get the distance and angle (with respect to the robot orientation) of the obstacle.

LIDAR is relatively noise-free compared to most other sensors and helps in getting a more robust reconstruction of the environment.



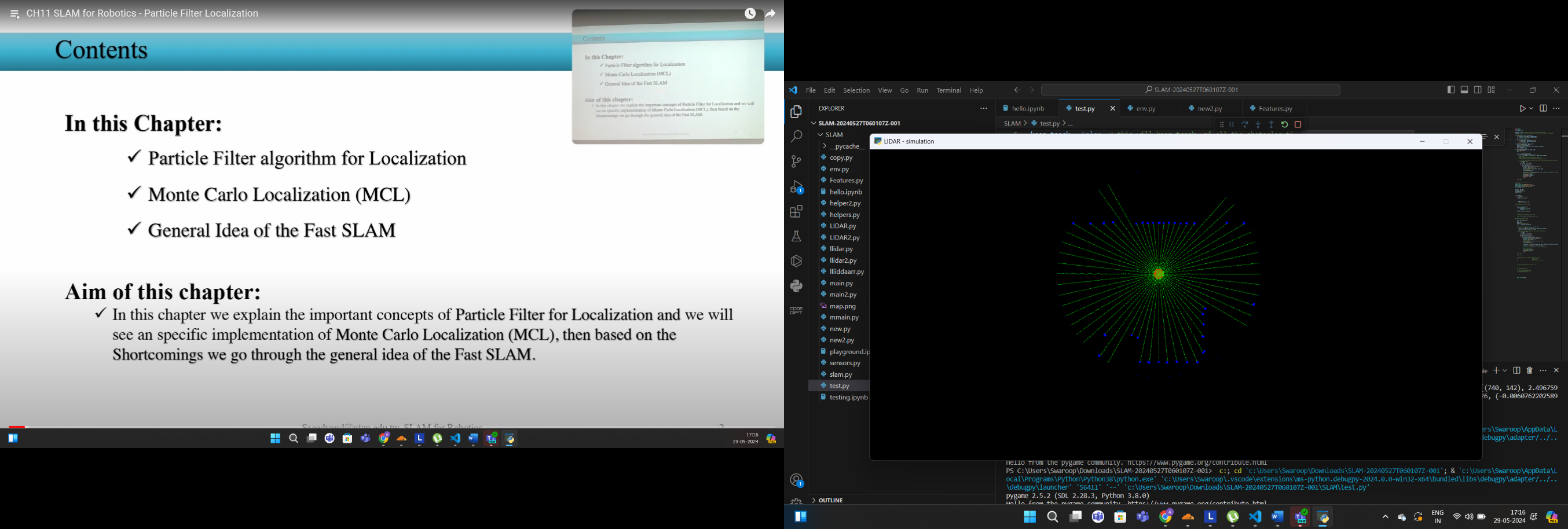


Fig: (above) The map of the environment the robot is in.

(below) representation of what the LIDAR data

Perceiving the environment –

**Localization problem** - It’s the task of estimating the position of the robot on a given map using robot's current and past sensory inputs.

Reconstructing an entire building with lidar SLAM : <https://youtu.be/08GTGfNneCI?si=qJFTsh-XO2ixcMQ_>

Orb slam <https://www.youtube.com/watch?v=UVb3AFgabu8>

Visual slam <https://youtu.be/9cPqbtiGWKM?si=O3Wr-kdVgCiMez4O>

Cnn based slam <https://www.youtube.com/watch?v=z_NJxbkQnBU>

The best way that I know, hands down, is by reading the book Probabilistic Robotics by Thrun, Burgard and Fox. This book starts with the very basics of Probability theory and builds on to SLAM. You'll learn everything from Particles filters to Optimization techniques in this book. It's amazing!

If you're not into reading a textbook, there are a few YouTube playlists available.

1. SLAM by Claus Brenner: [SLAM Lectures](https://youtube.com/playlist?list=PLpUPoM7Rgzi_7YWn14Va2FODh7LzADBSm&si=AiWBEyRsjCNSnmIW) . The best part about this playlist is that a full python based 2D SLAM is included. You get an assignment at the end of most videos to implement what you learnt on a python template that's already prepared. Great for hands on experience.
2. SLAM by Cyrill Stachniss: [SLAM Course (2013)](https://youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_&si=8E1VuZjn5ASTyBFe) . If you're into classroom-type courses then this might be good for you. The professor teaching the course is one of the most well known in the field of robotics. You'll definitely enjoy it!

Obviously, it's not limited to this. There is a crazy amount of papers published on SLAM. You just need to start looking. You can find papers which are complete tutorials for the type of SLAM you want to implement.

You can also look at the codebase of ROS2 Slam toolbox package. It's a great 2D SLAM algorithm to have a look at which will help you see how production level SLAM is programmed.

If you liked this answer, please do upvote! :)

The Quora thread on slam <https://www.quora.com/What-is-the-use-of-sparse-mapping-over-dense-mapping-in-Visual-SLAM>

Types of slam <https://www.linkedin.com/pulse/type-slam-robotics-prakash-thangavel/>