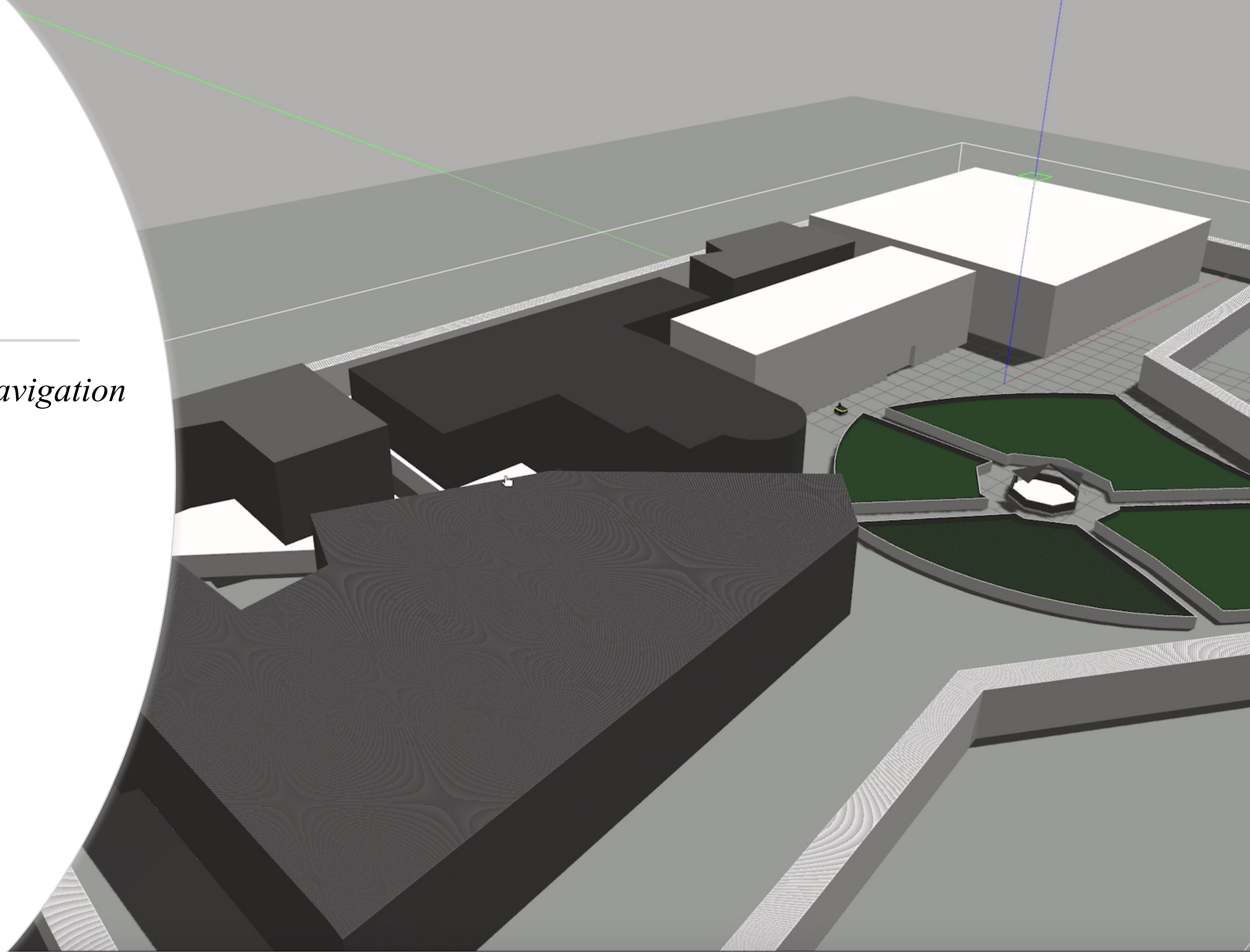
ROBOTICS PROJECT 2

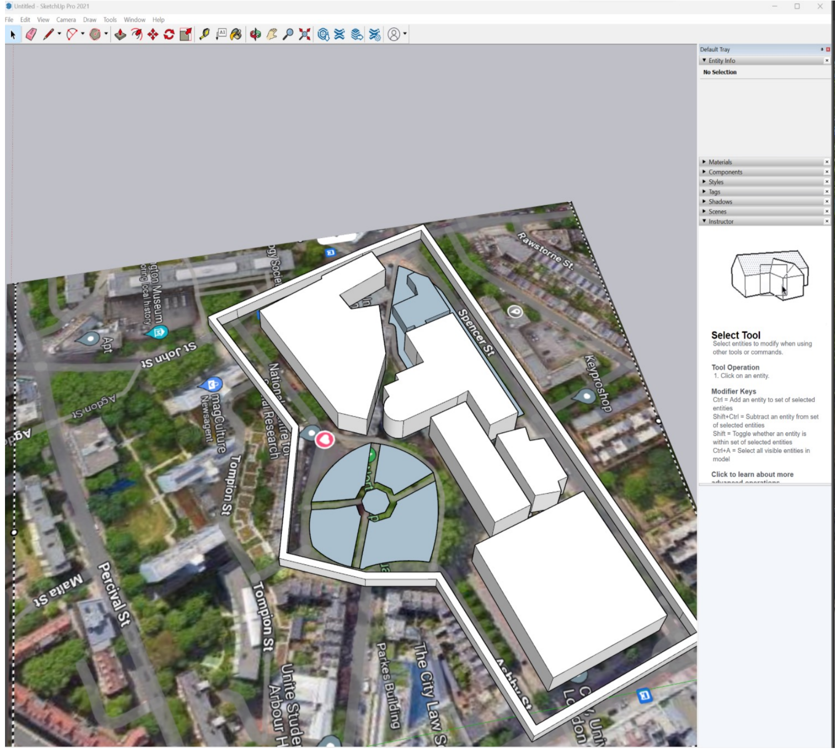
Robotic Simulation and Navigation



Prepared and Presented By:

- *Will Walaa*
- *Sheshta Ramgoolam Sobun*
- *Sorna Raj*
- *Pravesh Gnana Prakasan*
- *Srija Chakraborty*

City, University Campus World



```
export JACKAL_LASER=1
roslaunch unicampus unicampus.launch
```

Gazebo and Jackal Robot

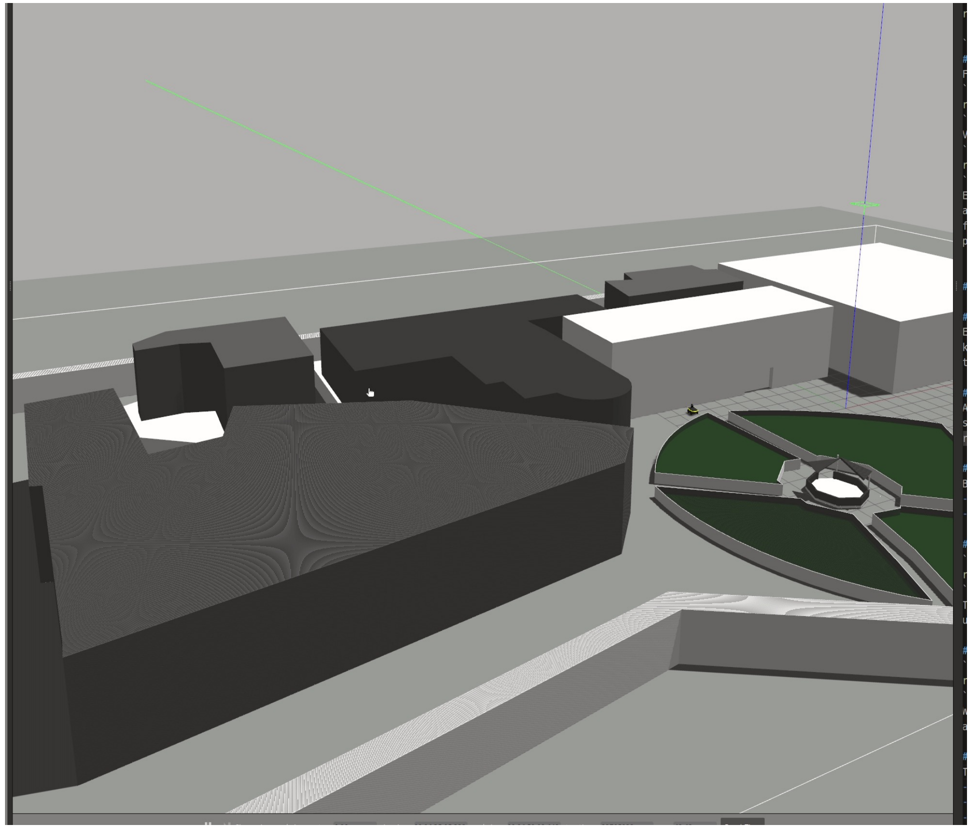
Gazebo

- Advanced Simulation Software
- Robust physics modelling
- High Quality Graphic
- Variety of sensors and objects

GAZEBO

Jackal Robot

- 4 Wheeled Robot
- Known for agility and versatility
- Front Laser Scanner Sensor
- Vital sensor that enable robot to perceive and interact with the environment effectively.
- Crucial for obstacle avoidance, navigation and mapping



SLAM PROBLEM FORMULATION

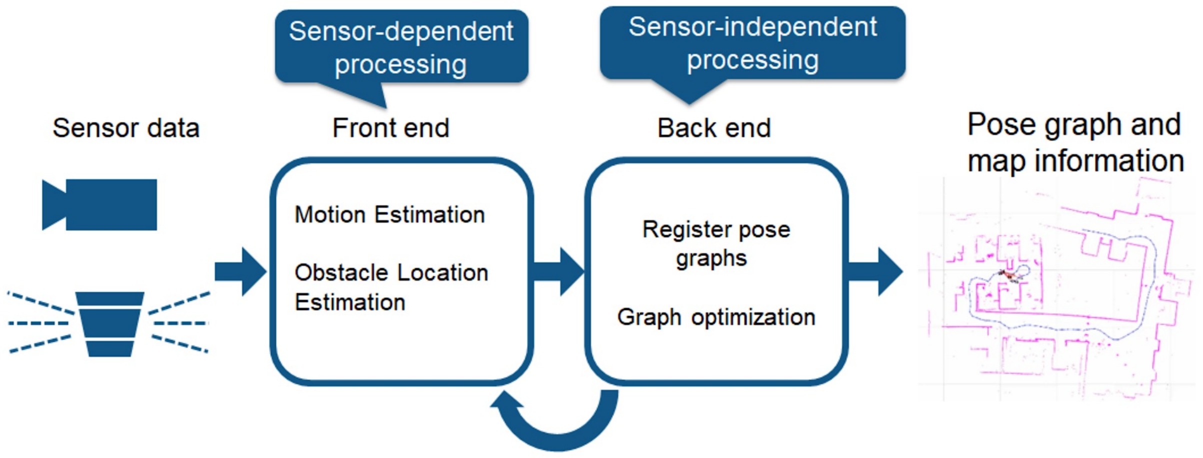
Consider a home robot vacuum. Without SLAM, it will just move randomly within a room and may not be able to clean the entire floor surface. In addition, this approach uses excessive power, so the battery will run out more quickly. On the other hand, robots with SLAM can use information such as the number of wheel revolutions and data from cameras and other imaging sensors to determine the amount of movement needed. This is called localization. The robot can also simultaneously use the camera and other sensors to create a map of the obstacles in its surroundings and avoid cleaning the same area twice. This is called mapping.

Gmapping using SLAM

- **Gmapping** is a package that provides laser-based SLAM as a ROS node. It is implemented based on Rao-Blackwellized particle filters.
- Takes data from both laser sensor and robot pose to build a 2D Map
- The input data it mainly needs is the raw laser scan, that provides distances and odometry.

HOW SLAM WORKS

Two types of technology components used to achieve SLAM.
Sensor signal processing, including the front-end processing, which is largely dependent on the sensors used.
Pose-graph optimization, including the back-end processing, which is sensor-agnostic.



SLAM Processing Flow

Simultaneous Localisation And Mapping (SLAM)

Data Collection: The robot's sensor reads that it is approximately 5 meters from a wall.

Bayesian Probability Update: Initially, the robot estimates its distance from the wall with some uncertainty. When it receives the sensor data (5 meters), it updates this estimate. If its prior belief was 4.5 meters, the new data shifts this belief closer to 5 meters.

Gaussian Distribution Modeling: The robot models this new estimate using a Gaussian distribution, centered around 5 meters, with a variance that reflects the uncertainty (e.g., ±0.5 meters).

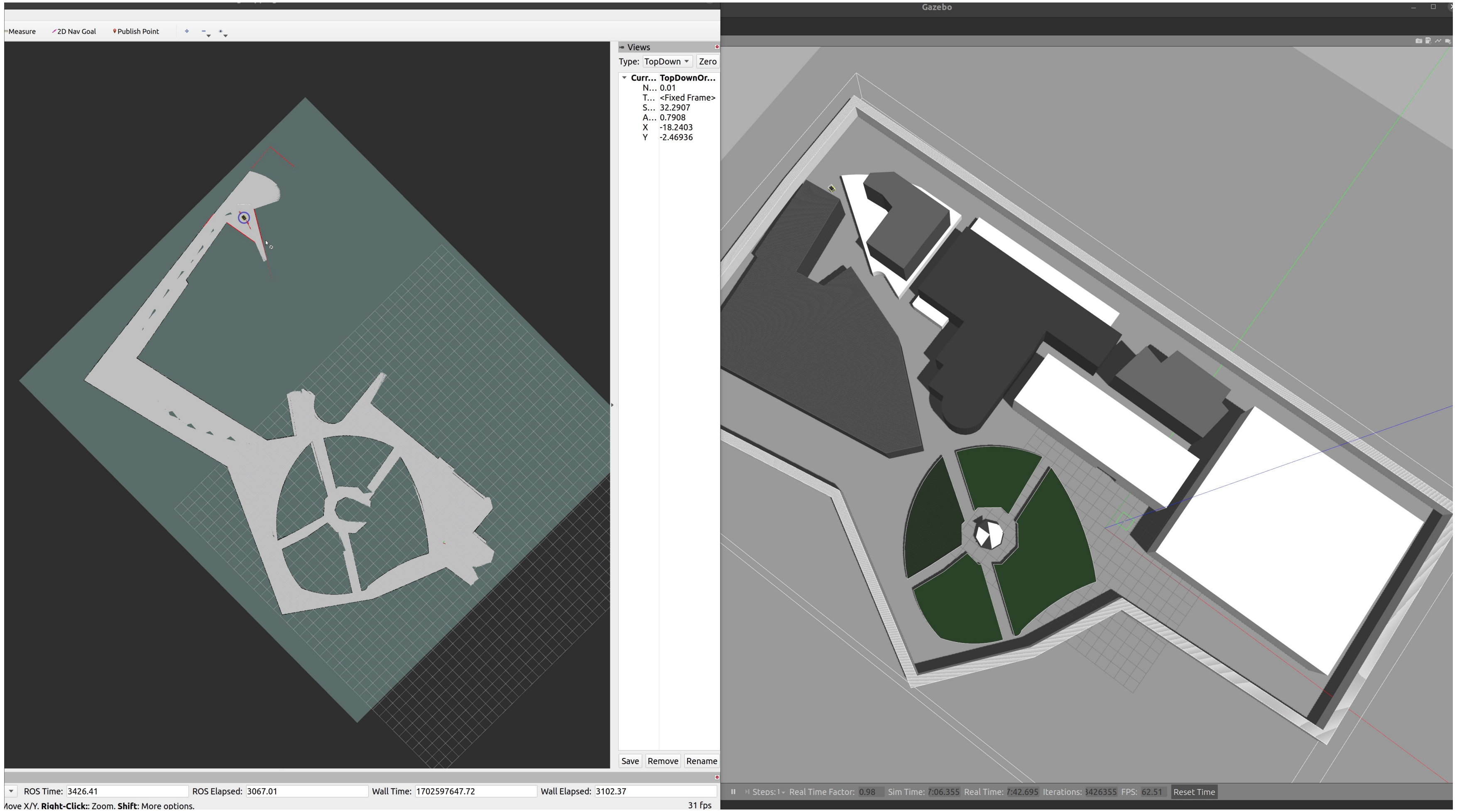
Kalman Filtering (If Applicable): The Kalman filter takes this Gaussian distribution and the new measurement, and updates the estimate to reduce the variance, increasing confidence in the robot's position.

Particle Filter Application: In particle filtering, multiple 'particles' representing different estimated positions are adjusted. Each particle might initially indicate positions between 4 and 6 meters. As the sensor reads 5 meters, particles closer to this value become more probable.

Graph-Based SLAM Optimization: In graph-based SLAM, this measurement creates a new node (representing the robot's position) and an edge (its relation to the wall), which is then optimized in the overall map graph for consistency with other measurements.

Output: The robot now has a refined estimate of its position relative to the wall and updates its internal map of the environment accordingly.

Gmapping using SLAM



Keyboard Control - Teleop

Update Your Package List

First, make sure you have the latest information about available packages:

```
sudo apt-get update
```

Install the teleop_twist_keyboard Package

Depending on your version of ROS, use one of the following commands:

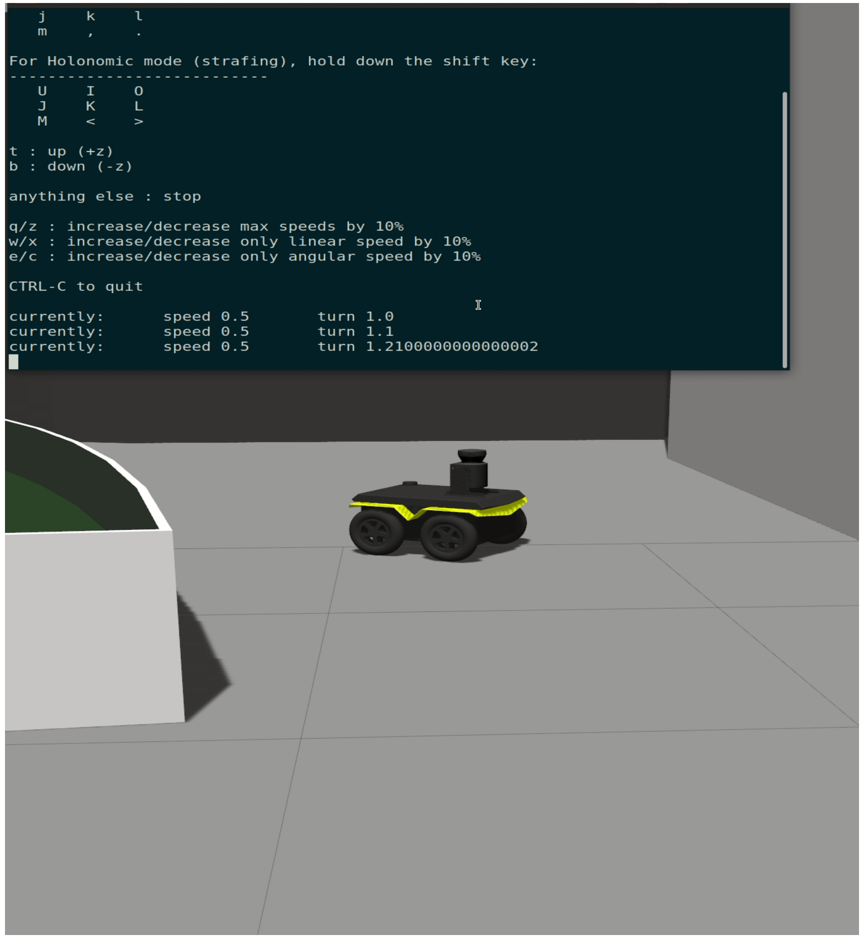
- For ROS Melodic or Noetic:

```
sudo apt-get install ros-noetic-teleop-twist-keyboard
```

Using the teleop_twist_keyboard Package

Once the installation is complete, you can launch the keyboard teleoperation interface:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```



Adaptive Monte Carlo Localisation (AMCL)

Initial Setup: Imagine the robot starts in a room, unsure of its exact location. It creates a cloud of 1,000 particles, each representing a possible position.

Sensor Reading: The robot's sensor reads a wall 3 meters ahead.

Weighting Particles: Particles closer to being 3 meters from a wall receive higher weights, as they match the sensor reading.

Resampling: The system resamples, giving preference to higher-weight particles. Particles far from any wall are likely removed.

Motion Update: The robot moves forward by 1 meter. Each particle also moves forward by approximately 1 meter, introducing some variation to account for uncertainty in movement.

Repeat with New Data: As new sensor data comes in the process repeats, continually refining the particle cloud.

Output: Gradually, most particles converge to the robot's actual location, providing a reliable estimate of its position.

The Kullback-Leibler (KL) divergence is a mathematical concept from information theory that measures how one probability distribution diverges from a second, reference probability distribution. It's often used in contexts like machine learning, statistics, and signal processing to compare distributions

For two discrete probability distributions P and Q defined on the same probability space, the KL divergence from Q to P is defined as:

$$D_{KL}(P||Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

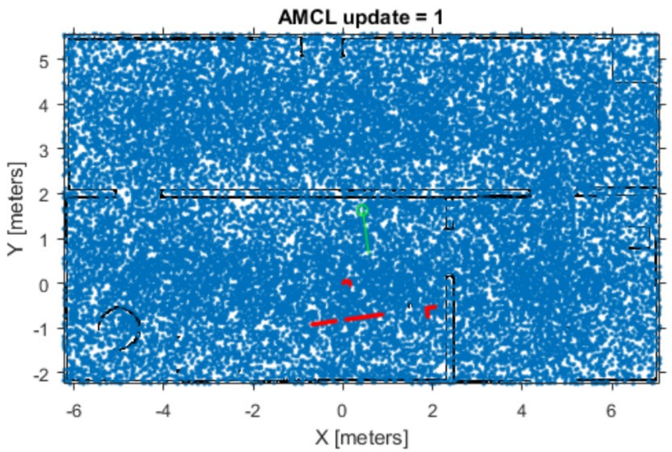
For continuous probability distributions, the summation is replaced by an integral:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

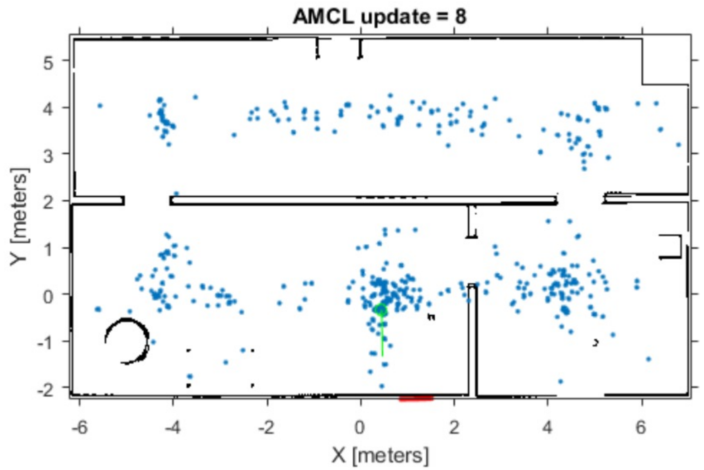
where $p(x)$ and $q(x)$ are the probability density functions of P and Q, respectively.

Adaptive Monte Carlo Localisation (AMCL)

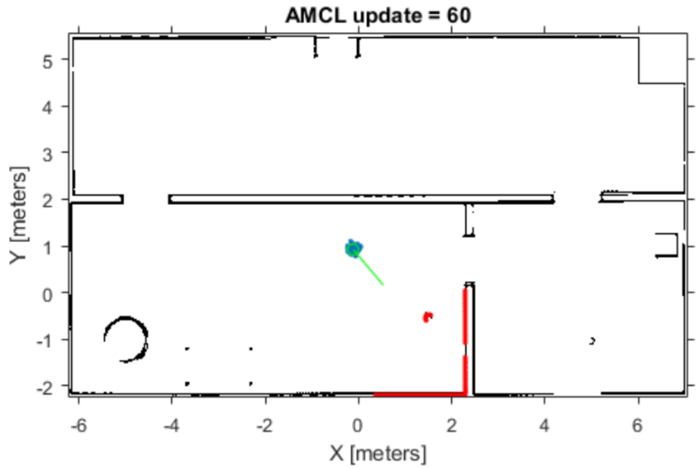
After first AMCL update, particles are uniformly distributed



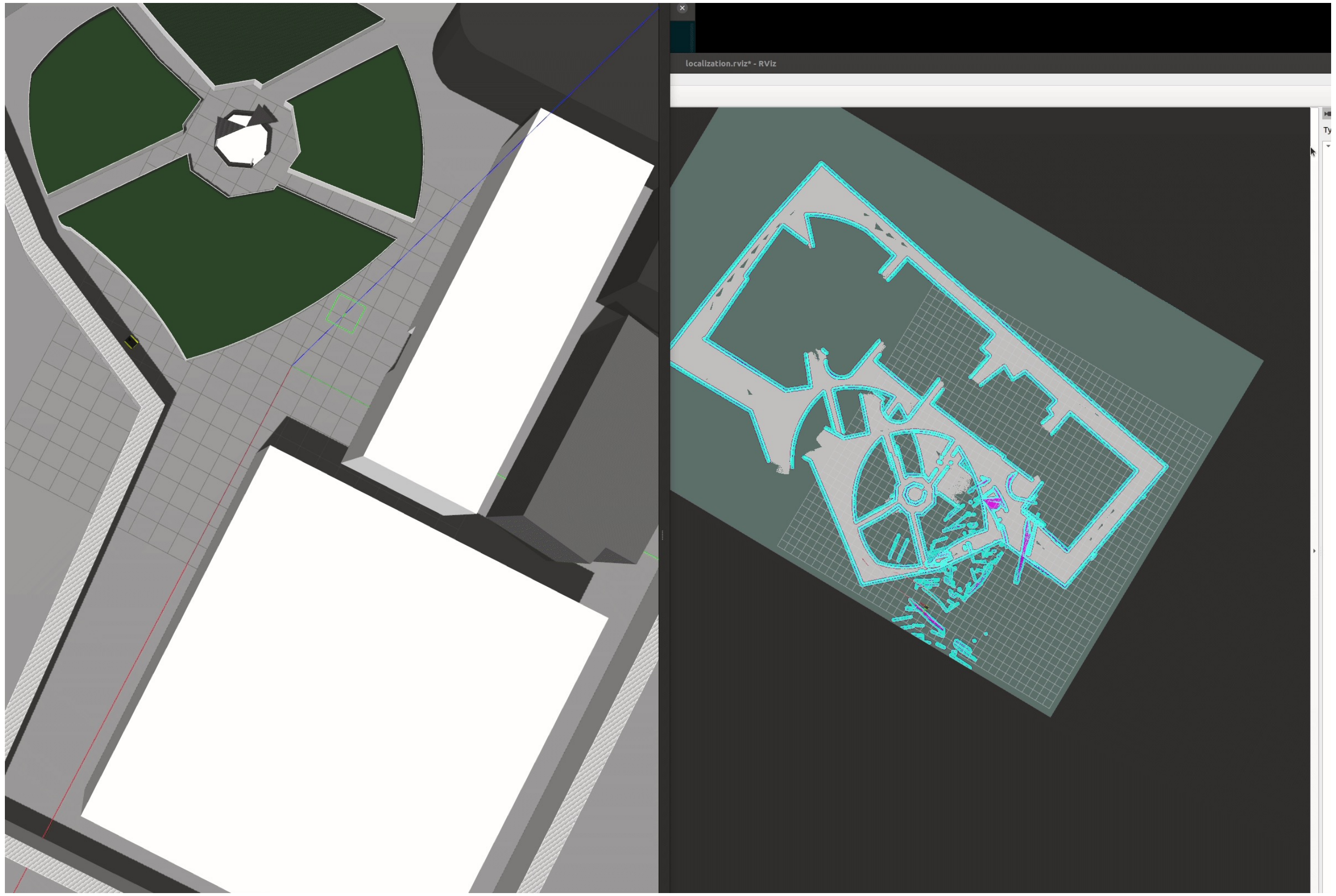
After 8 updates, the particles start converging to areas with higher likelihood



After 60 updates, all particles should converge to the correct robot pose and the laser scans should closely align with the map outlines.



Adaptive Monte Carlo Localisation (AMCL)



Challenges, Improvements & Skills Gained

Challenges

- ❑ Ensuring accurate mapping in the environment
- ❑ Accurate mapping in dynamic environment where obstacles and condition change
- ❑ Dealing with sensor noise and data inconsistency

Skills Gained

- ❑ Simulation and Modeling in Gazebo
- ❑ SLAM principles using Gmapping package
- ❑ Robot Localisation using AMCL for accurate robot positioning
- ❑ Path Planning and Navigation
- ❑ Sensor Data Analysis
- ❑ ROS for robotic programming and system integration
- ❑ Team Collaboration and Project Management skills

Improvements

- ❑ Data Filtering to smooth out noisy sensor data. Kalman filters predict the state of the system and then correct it with the sensor input thus improving accuracy
- ❑ Sensor Fusion to combine data from multiple sensors e.g using LiDAR
- ❑ Algorithmic Improvement to optimizing the SLAM algorithm and path planning to make it more robust against noise and errors
- ❑ Machine Learning techniques to predict and compensate for sensor inaccuracies. For example, using neural networks can be trained on large datasets and correct anomalies on the sensor data

Clear Instructions for Students & New learners Realistic Simulation || Added Challenges||Professional ReadMe File

https://github.com/wlaa41/clearpath_university-campus