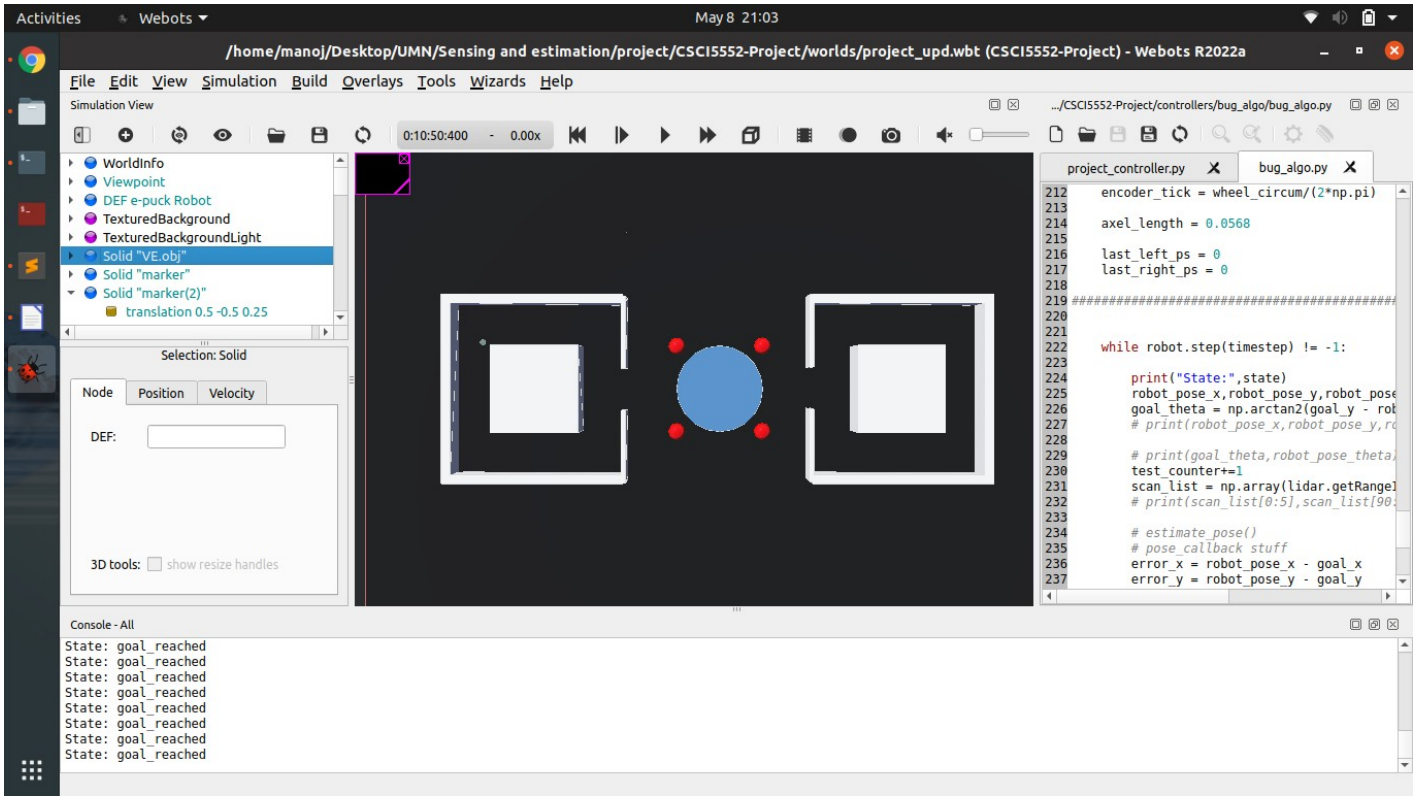
CSCI 5552 – Sensing and Estimation in Robotics

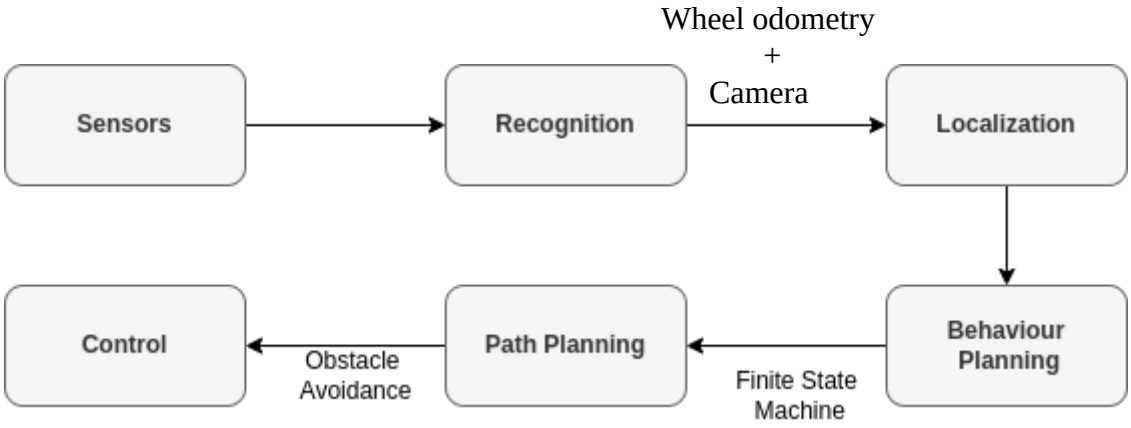
Team (Individual): Kavish Manoj Shah
UMN ID: 5732010
PROJECT: Localization for Navigation
OPTION: With Landmarks

Objective:
As part of the project, we are provided with a simulation environment in Webots. The aim of the project is to develop SLAM algorithm which would navigate the robot from a known starting location to the specified destination autonomously. Below is the environment in the Webots simulator.



The robot provided for the navigation task is the E-Puck, which is a differential drive robot with various sensors such as Lidar, Camera, Wheel Encoders, Gyroscope and position sensors.
Navigation task requires us to do various sub-tasks such as Sensing the environment, localizing the robot in the environment, path planning and controlling the movement of the robot. All these tasks are performed simultaneously in order as all the mentioned tasks required information from each other. For path planning I have used the Bug-2 algorithm.
The project performed by me uses Lidar, Camera and position sensors as the primary sensors to solve the navigation task. In real world applications most of these sensors are noisy. In his case, wheel encoders/ position sensors are noisy.

Navigation Workflow:



Sensors:

Lidar:

A typical lidar sensor emits pulsed light waves into the surrounding environment. These pulses bounce off surrounding objects and return to the sensor. The sensor uses the time it took for each pulse to return to the sensor to calculate the distance it traveled. Lidar can be of various types, 1D lidar, 2D and 3D.

1. 1D Lidar: sensors work quite similarly to ultrasonics, but use light instead of sound.
2. 2 D Lidar: these lidars have a rotating disc which in turn has a laser attached to it. 2D LiDAR sensors (2D laser scanners as well) are suitable for performing detection and ranging tasks on surfaces.
3. 3D Lidar: as opposed to 2D lidars, 3D lidars captures volume instead of planes. These lidars have multiple chanels which give the Z-axis. These type of lidars nearly capture the whole enviroenment but are computationally expensive.

In my case, I've used 1 single layer of the lidar(2D Lidar) with a horizontal resolution of 180 and field of view of 3.14 radians. As seen below, the blue colored projection are the lidar points being projected onto the walls of the environment. In addition to this, there can be seen an empty path in the middle which indicates that the lidar does not see an obstacle in its range and is free path for movement.

Position Sensors:
The position sensors or wheel encoders are used to estimate wheel odometry. Position sensors are located directly behind each motor is a wheel encoder. Each wheel encoder is used to count the number of times the motor (left or right) has rotated. This can be used to calculate the distance that the robot has driven or turned. The wheel encoder provides us the encoder ticks and the current pose of the robot can be calculated as illustrated below:

Velocity Equations:

$$V_{left} = (v - \omega L)*0.5$$
$$V_{right} = (v + \omega L)*0.5$$

Linear Velocity : $V = (Distance_{left} + Distance_{right})*0.5$

Angular Velocity : $\omega = (-Distance_{right} + Distance_{left})*L$

Update Equations:

$$x' = x + v * cos(\theta) * \Delta t$$
$$y' = y + v * sin(\theta) * \Delta t$$
$$\theta' = \theta + \omega * \Delta t$$

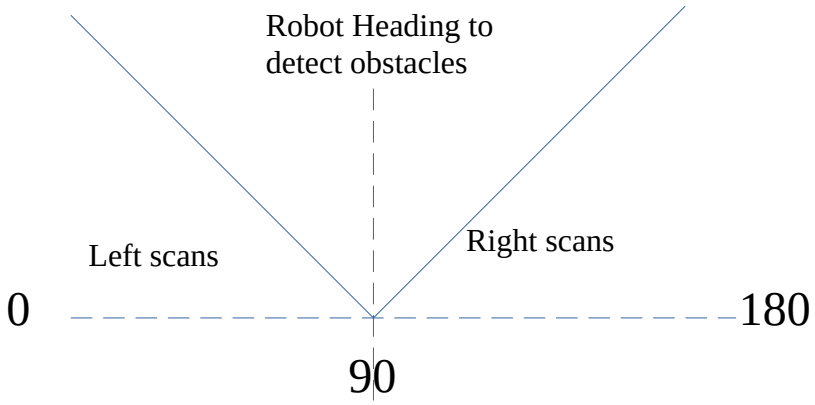
Camera:

Camera is used to recognize the webots objects spawned in the environment. I spawned 4 balls (spheres) in the environment and the object recongnition function inbuilt in webots gives the poses of landmark in global frame. This can further be used in order to calculate transformation matrices which further gives us the pose of robot in global coordinates. I further used the global pose from the camera recognition to update the noisy pose from wheel odometry. The updated pose helps in correcting the robot pose and hence improve localization. Below is a screenshot of the camera frame from the webots simulator.

Behavior and Path Planning:

In robot navigation, path planning plays a crucial role. The primary aim of path planning is to generate a collision-free path for the robot from the start position to the goal position. In this project, I used the Bug-2 path planning algorithm. In this algorithm, we need to know the current and goal poses of the robot. Below is the Bug-2 algorithm and pseudocode.

For the Bug-2 algorithm, we would slice the the lidar scans

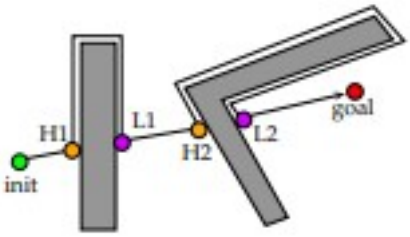


repeat until goal is reached

- head toward goal
- if sensor reports contact with an obstacle then
  - follow the obstacle until it encounters the *m*-line again
  - leave the obstacle and continue straight toward goal

Bug2 Pseudocode

1: $L_0 \leftarrow$ init; $i \leftarrow 1$
2: **loop**
3: **repeat** move on a straight line from L_{i-1} to goal
4: **until** goal is reached or obstacle is encountered at H_i
5: **if** goal is reached **then** exit with success
6: **repeat** follow boundary
7: **until**
(a) goal is reached or
(b) *m*-line is re-encountered at Q such that $Q \neq H_i$, $d($
line $(Q,$ goal$)$ does not cross the current obstacle a
9: **else if** H_i is re-encountered **then** exit with failure
10: **else** $L_i \leftarrow Q$; $i \leftarrow i + 1$



References:

[1] http://www.cs.columbia.edu/~allen/F17/NOTES/icckinematics.pdf

[2] https://www.ri.cmu.edu/pub_files/pub1/dellaert_frank_1999_2/dellaert_frank_1999_2.pdf

[3] https://cs.gmu.edu/~ashehu/sites/default/files/cs485_Fall2013/ShehuLecture02.pdf

[4] Devansh Verma, Priyansh Saxena, Ritu Tiwari, Robot navigation and target capturing using nature-inspired approaches in a dynamic environment

[5] https://www.youtube.com/playlist?list=PLbEU0vp_OQkUab-znh3nej4o49hxoMfZu