ENAE450 Final Project Report

Brandon Newman, Sriman Selvakumaran, Adithya Sundar, Joshua Stone

Due: May 18th, 2024

1 Introduction

The following report is for our group's final project for ENAE450. The final project required us to navigate a maze using the Turtlebot3 Waffe Pi.

We utilized ROS2 and Python (`rospy`) to program the navigation code, which utilzed the turtlebot's lidar sensor and camera.

The navigation code was tested using the physical robot (See section 3), as well as simulated Gazebo runs (See section 2).

2 Simulation

2.1 Methods

2.1.1 Introduction

Our team utilized ROS2, Python, and Gazebo to simulate the navigation for the Turtlebot3 Waffle Pi.

2.1.2 Navigation Code

2.2 Results

3 Hardware

3.1 Methods

3.1.1 Introduction

Our team used ROS2 and Python to implement the following navigation algorithm on the Turtlebot3 Waffle Pi. The code utilizes the Lidar sensor and the front camera.

3.1.2 Navigation with Lidar

To start, our navigation ROS node subscribed to the Lidar sensor data topic (`/scan`), and created a publisher for the `/cmd_vel` to move the turtlebot.

```
self.scan_subscriber = self.create_subscription(
    LaserScan,'/scan',self.scan_subscriber_handler, 10)
self.cmd_vel_publisher = self.create_publisher(
    Twist, '/cmd_vel', 10)
```

3.1.3 Aruco Marker Detection

Additionally, our team used the turtlebot's front facing camera to detect Aruco markers. This was done as part of the bonus points for the project. For Aruco marker detection, our team used the opencv library to analyze the camera frame from the turtlebot. We created another ROS node to perform Aruco detection, communicating with the navigation node via a service. Both nodes are spawned from the launch file.

First, we import opencv (`cv2`) and define the aruco detection variables. We also import the `Image` class for the camera subscriber, and the custom service message `ArucoDetectSrv` for the service client:

```
import cv2
from sensor_msgs.msg import Image

dictionary = cv2.aruco.getPredefinedDictionary(cv2.aruco.
    DICT_4X4_1000)
parameters = cv2.aruco.DetectorParameters()
detector = cv2.aruco.ArucoDetector(dictionary, parameters)
```

Next, we created a subscriber to the `/image_raw` topic to get the camera data from the turtlebot. We also create a publisher to the aruco detection service.

```
self.scan_subscriber = self.create_subscription(
    Image,'/image_raw',self.frame_handler,10)
self.aruco_client = self.create_client(
    ArucoDetectSrv, '/aruco_detected')
```

In the subscriber callback function, we use the open cv library to detect arucos on the frame:

```
def frame_handler(image_data, self):
    # get camera frame data
    frame = image_data.data

    # detect arucos
    (corners, ids, rejected) = detector.detectMarkers(frame)
```

Then, we send these results to the navigation script via service call:

```
# if ids detected, publish to client
if len(ids) > 0:
    req = ArucoDetectSrv.Request()
    req.corners = corners
    req.ids = ids

    self.aruco_client.call_async(req)
```