COSC-5207EL02: Assignment 3 Report

Group Number: 8

Group Member:

NAME	STUDENT#	EMAIL
Haoliang Sheng	0441916	hsheng@laurentian.ca
Songpu Cai	0441024	scai1@laurentian.ca

Introduction

This project focuses on enhancing the autonomous navigation capabilities of the TurtleBot3 robot within a customized simulation environment in Gazebo. Through the development of a specialized ROS2 package and modifications to the simulation world, the project aims to address the challenges of navigating through an environment with obstacles.

- **turtlebot3_move_cam.py**: This Python script is the core of the autonomous navigation program, utilizing ROS2 and OpenCV to process images from the TurtleBot3's camera. It detects obstacles and walls by color, adjusting the robot's trajectory to avoid collisions.
- **setup.py**: Located in the navigation_pkg package, this file is modified to include turtlebot3_move_cam.py as an executable ROS2 node, facilitating its integration into the ROS2 ecosystem and enabling easy execution of the navigation program.
- **custom_turtlebot3_world.world**: A customized Gazebo world file that incorporates multiple red cube obstacles to simulate a more challenging navigation environment. This world tests the TurtleBot3's ability to detect and avoid obstacles.
- **custom_world.launch.py**: A custom launch file that initializes the Gazebo simulation with the custom_turtlebot3_world.world. It ensures the autonomous navigation system and the TurtleBot3 robot model are correctly loaded and

integrated for simulation.

- **model.sdf**: Defines the specifications of the custom red cube obstacles added to the Gazebo world, including their dimensions, color, and physical properties. It is crucial for replicating realistic obstacles within the simulation environment.

For a comprehensive understanding of the implementation and its results, a YouTube video demonstration will be provided https://www.youtube.com/watch?v=WYiP-SY0vJI.

2. Customize the Gazebo World

2.1 Overview

The task enhances the Gazebo world for TurtleBot3 by adding five red cube obstacles to simulate navigation challenges. This aims to create a realistic testing environment for autonomous navigation and obstacle avoidance algorithms. A custom launch file is also developed to integrate these changes, facilitating the simulation of complex environments for robotic testing and development.

2.2 Methodology

Adding red-colored 5cm cube obstacles

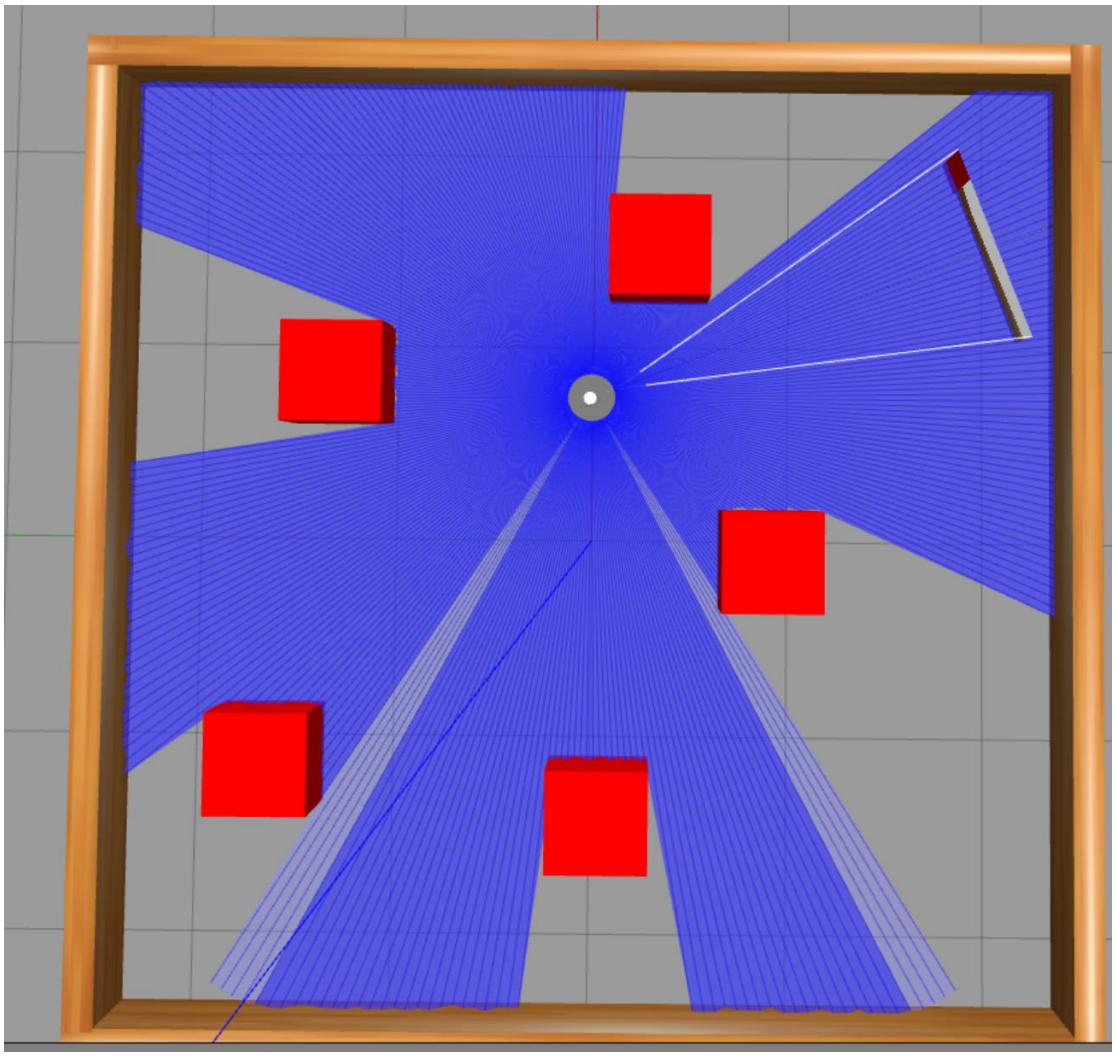
- Utilized Gazebo for simulation setup and environment customization.
- Selected the "**Box**" shape from the menu and dragged it into the panel to start the obstacle creation process.
- Right-clicked the box and selected the "**Edit Model**" button, then clicked on the box again and chose "**Open Link Inspector**" to adjust properties.
- Changed the "**Visual**" and "**Collision**" geometry settings by modifying the size to create a cube with each side measuring 50cm.
- Adjusted the "**Visual**" properties' "**Ambient**" and "**Diffuse**" colors to red to mimic the desired obstacle appearance.
- Saved the customized unit_box to the **model_editor_models** directory, which automatically generated a **model.sdf** file, defining the model's specifications.
- Saved the modified world as **custom_turtlebot3_world.world** in the specific

directory:

workspace/src/turtlebot3_simulations/turtlebot3_gazebo/worlds/, ensuring the custom obstacles were included in the simulation environment.

Custom Launch File Creation

- Located the existing launch file **turtlebot3_dqn_stage1.launch.py** in the **workspace/src/turtlebot3_simulations/turtlebot3_gazebo/launch/** directory. Modified the file to load the custom world by updating the **os.path.join** method to point to the new **custom_turtlebot3_world.world** file.
- Save the py file as **custom_world.launch.py**



These steps enabled the creation of a customized Gazebo world filled with red cube obstacles designed to challenge and test the TurtleBot3's autonomous

navigation capabilities. The modification of the launch file ensured that the simulation environment is easily accessible and integrated with the necessary navigation systems for comprehensive testing.

3. Autonomous Navigation Program

3.1 Overview

The Autonomous Navigation Program leverages ROS2 and OpenCV to enable the TurtleBot3 Waffle Pi to navigate a customized Gazebo world, avoiding collisions with red cube obstacles and walls through camera-based obstacle detection. The program subscribes to the robot's camera feed, using color detection in OpenCV for real-time obstacle recognition. It then dynamically steers the robot away from potential hazards, focusing solely on visual data for navigation. This approach tests the effectiveness of image processing in autonomous navigation, aiming for a robot that can adaptively and safely move through its environment without relying on lidar or other sensors.

3.2 Methodology

Creating the Package:

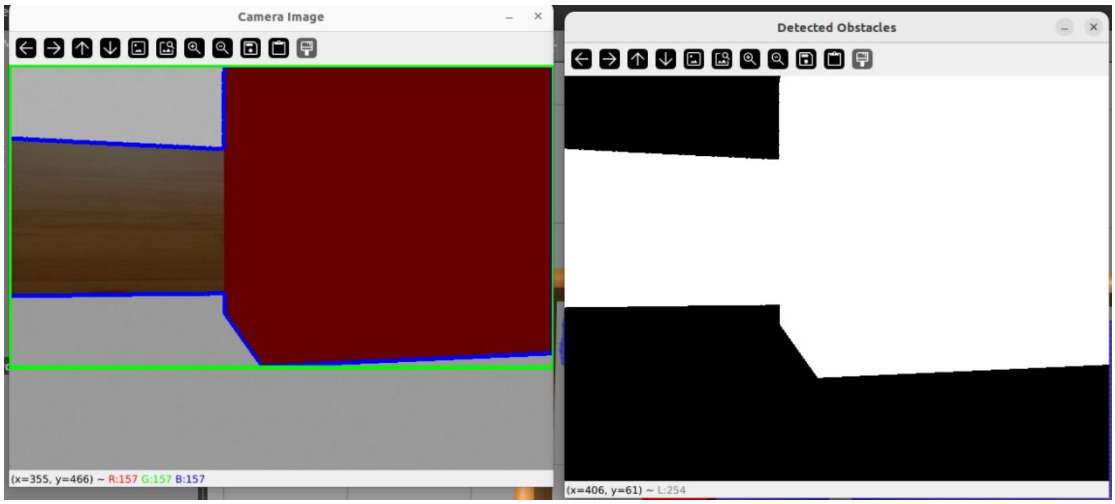
- Navigate to the workspace source directory **workspace/src/** and create a new ROS2 package named **navigation_pkg** using the command: `**ros2 pkg create --build-type ament_python navigation_pkg**`
- Copy the **turtlebot3_move_cam.py** script into the **workspace/src/navigation_pkg/navigation_pkg** directory to include it in the new package.
- Modify the **setup.py** file located in **workspace/src/navigation_pkg/**. In the **entry_points** section under **console_scripts**, add the following line to ensure the turtlebot3_move_cam.py script is executable as a ROS2 node: `**'turtlebot3_move_cam = navigation_pkg.turtlebot3_move_cam:main',**`.

Navigation Logic:

- Modify the **image_callback** function within the **turtlebot3_move_cam.py** script. Utilize HSV color space and **cv2.inRange** to filter out the colors of red obstacles and the wooden texture of walls.
- Combine the masks for the red and wood colors, and find the largest contour to wrap with a bounding rectangle.
- Calculate the height of this bounding rectangle. If the height exceeds 350 pixels, **set self.state** to "**turn**". Otherwise, continue with "**move forward**".
- The wooden texture for walls is identified based on the **custom_turtlebot3_world.world** file, and the HSV range for wood can be extracted from the image found at **/usr/share/gazebo-11/media/materials/textures/wood.jpg**.

Modifying the Launch File:

- Open the previously created **custom_world.launch.py** in the **workspace/src/turtlebot3_simulations/turtlebot3_gazebo/launch/** directory.
- Import the Node class from launch_ros.actions: **from launch_ros.actions import Node**.
- Create a **turtlebot3_move_cam_node** and return a **LaunchDescription** class that includes this node for executing the navigation program within the simulation environment.



These steps collectively define the process of setting up the autonomous navigation program for the TurtleBot3 robot. By focusing on image processing to detect

obstacles and modifying the robot's state based on the visual cues, this methodology aims to achieve effective autonomous navigation without the need for advanced sensors like lidar. The creation of a dedicated package and the adjustment of the launch file ensure that the navigation logic is seamlessly integrated into the simulation environment, allowing for real-world testing and development.