**Senior Design**

ENG EC 463

To: Professor Pisano

Team: 10

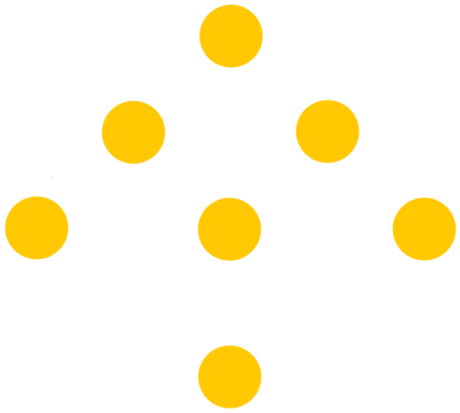
Date: 11/24/2024

Subject: First Prototype Testing Report

---

# First Prototype Test Plan

## Draper Convoy



By

Team 10

**Team Members**

Sadman Kabir: kabirs@bu.edu
Benjamin Hsu: bhsu25@bu.edu
Nikhil Krishna: nikhilk@bu.edu
Hugo Silmberg: hsilmber@bu.edu
Michael Moran: mmoran3@bu.edu

## Required Materials

Hardware:

- Desktop computer running Ubuntu and access to Gazebo simulation software

Software:

- Gazebo software
    - Simulates TurtleBot sensor information and output
- ROS2 installed on the desktop computer
    - Using ROS2 Humble distribution
    - Allows data to be accessed through TurtleBot sensors and communicate with other software
- Person detection node
    - YOLO object detection algorithm
        - Pretrained model from Ultralytics
    - Identifies human beings detected within camera view and outputs information about their position relative to the robot
    - Highly accurate, even from long distances and figure partially cut off
- Motor control node
    - Accepts data from the person detection node and outputs information to control the TurtleBot’s motors

## Set-Up

1. Open a terminal window

2. Enter this in to source it correctly:

   -
     ```
     cd ~/ros2_ws
     ```

   -
     ```
     source /opt/ros/humble/setup.bash
     ```

   -
     ```
     source install/setup.bash
     ```

3. To launch the simulation enter this:

   -
     ```
     ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py
     ```
   - Want to launch the simulation with nav2, slam and, rviz? Enter this:

     -
       ```
       ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py nav2:=true slam:=true rviz:=true
       ```
   - Want to launch the simulation with a preexisting map for nav2:

     -
       ```
       ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py nav2:=true slam:=true rviz:=true
       ```

4. Open another terminal window and source it like in **step 2**

   - Enter this to launch the motor control node:

     -
       ```
       ros2 run my_teleop_pkg teleop_twist_keyboard
       ```

5. Open another terminal window and source it like in **step 2**

   - Enter this to launch the perception node for person detection:

     -
       ```
       ros2 run yolo_person_detector yolo_person_detector
       ```

6. When you get everything launched, make sure to unpause the simulation and then click on the terminal with the motor control to put it in focus so you can use the keyboard to control it

(Node files should be located in ~/ros2_ws/src.)

## Pre-Testing Set-Up Procedure

1. Ensure Gazebo simulation is running correctly.
2. Check that all nodes are running and responding.
3. Ensure control functionality is working correctly.

## Testing Procedure

- Place a human model in the simulation and turn the TurtleBot to identify the human.
- The person detection node identifies the human and sends data to the motor control node.
- The TurtleBot begins to move toward the human model.
- Move the human model around the simulation to demonstrate real-time identification and data processing.
- Move the human model outside of the camera's frame and test relocation functionality.
- Controls:
    - P: Autonomous driving mode + relocation
    - U I O; J K L; M , . ; Remote control driving
    - anything else: stop

## Discussion and Results

The measurements taken in this prototype testing were behavior based criteria for our TurtleBot. Since we do not have a physical bot yet we used Gazebo to test our progress so far. The behaviors we were looking for in this test were the following:

| **Functionality** |
|---|
| Identifies a human model |
| Bounding box around model w/ confidence level and centroid |
| Seek human target |
| Stop when close enough to the human model |
| Auto relocate -> continue seeking humans |

Our test successfully identified a human in the simulation seen by a bounding box that was placed around the human with the confidence of the identification. Our test showed that even from an extremely close distance we could identify a human as long as the camera had a view of the waist, any closer and the legs would not be identified as a human alone. For the seeking behavior of the bot it was also successful, able to move towards the human as long as the human was in view of the camera. The bot was able to consistently maintain a confidence >=80 while in seeking behavior. The bot successfully stopped near the human still within range of identification. Finally for Auto relocate, by moving the human around we showed that the robot will still follow the human as long as it stays in view. If we stop the human and then move the robot out of view, it successfully did up to a 360 degree turn to see the human once again and continued its seeking behavior.

From this test we can make a few conclusions of its current success and what still needs to be done. The correct seeking behavior showed that the human detection node correctly works to a basic degree, able to identify a human and follow it however if we put more humans in frame it could get confused, this will be solved in the next prototype by adding specific human identification. For the motor control aspect of this behavior we conclude that it works however is not fully optimized yet, the camera frame was oscillating to keep the human in view which should be stabilized and the stopping distance can be shortened to match the minimum distance of leader identification. As for obstacle avoidance, Gazebo was having trouble with showing the LiDARs data so we decided to not include this aspect in the prototype however this is a must do for our project and will be the most important improvement for our next prototype.