# **Assignment 3 (10%):**

**Autonomous Navigation with TurtleBot3 in a Custom Gazebo World**

DUE DATE: Friday, March 15th, 2024, 11:59 PM ET

**Objectives:**

* Understand the basics of ROS2, Gazebo simulation, and OpenCV image processing.
* Integrate and apply the vision module developed in the previous assignment for obstacle detection.
* Customize a Gazebo simulation environment for the TurtleBot3.
* Implement an autonomous navigation system that uses visual information to avoid obstacles.

**Assignment Guidelines:**

* All students in this course must read and meet the expectations described in the [Student Academic Integrity](https://intranet.laurentian.ca/policies/2017.09.19%20-%20Policy%20and%20Procedures%20on%20Academic%20Integrity%20-%20EN.pdf).
* Assignments must follow the programming standards document published on the course website in the D2L. Marks will be taken off if standards are not followed.
* **Submit just one (preferable) .py file AND the associate report in PDF per group**. Name the file based on your group “ID” and the assignment number, exactly as in this example for **assignment x and** **group x**: **CPSC\_5616EL\_Ax\_Gx.py**. Same naming convention applies to the PDF, **CPSC\_5616EL\_Ax\_Gx.pdf**.
* **Do NOT zip the files** that you submit.
* You may submit the assignment multiple times, but only the most recent version will be marked.
* **If applicable:** After the due date and time, a late penalty of 2% per hour, or a portion of an hour, will be applied. After 49 hours, the penalty reaches 100%, and submissions will no longer be accepted. The date and time of the last file submitted will determine the mark for the entire assignment. **These late penalties will not be applied under the circumstances of a cyber incident.**
* We compare all submitted assignments with one another, and pursue academic dishonesty vigorously. **You must complete the Honesty Declaration in the D2L before you will be able to submit your assignment.**

1. **Introduction**

This assignment builds upon your previous experience with ROS2, OpenCV, Gazebo, and the TurtleBot3 Waffle Pi, where you developed a vision module. Now, you will enhance your autonomous navigation system by incorporating the vision module you developed in the previous assignment. The focus will be on creating a custom simulation world and developing a program that enables the TurtleBot3 to autonomously navigate this environment, avoiding obstacles and walls using camera input.

**Requirements:**

* Completion of the previous assignment involving the development of a vision module with OpenCV.
* [ROS2 Humble, Gazebo simulator, TurtleBot3 simulation package](https://docs.google.com/document/d/1IKODn_HLyQuWZtbFf1y12FZc-qIcbTiybR_MVvwECGY/edit?usp=sharing).
* OpenCV Python library.

1. **Task 1: Customize the Gazebo World**

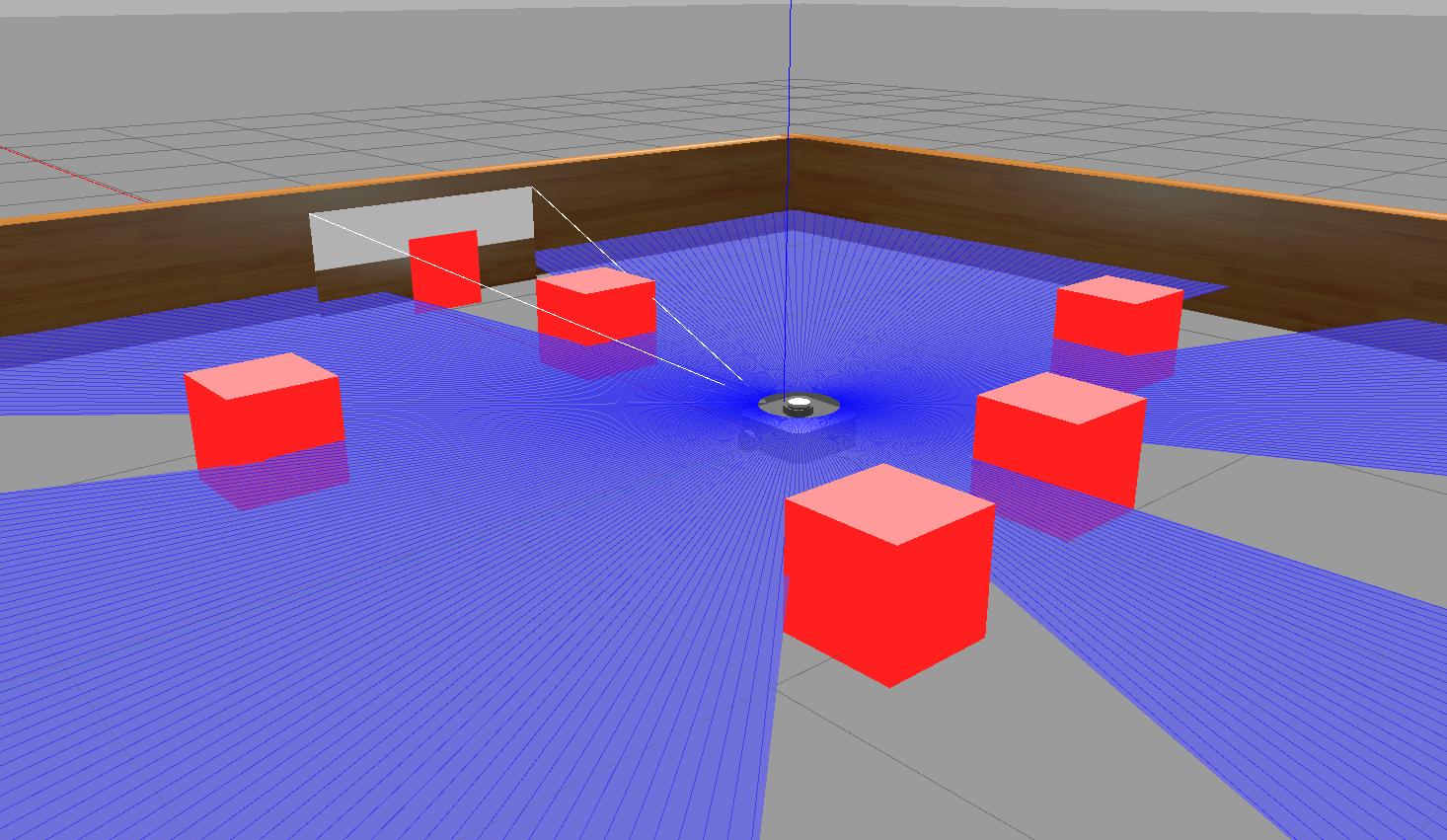


Figure 1

Create a customized version of the turtlebot3\_dqn\_stage1.world by adding multiple red-colored cube obstacles as shown in Fuigure 1.

**Instructions**:

* Launch Gazebo and open the turtlebot3\_dqn\_stage1.world.
* Add at least five red-colored cube obstacles at various locations within the world. Each cube should have a side length of 50cm.
* Save your custom world as custom\_turtlebot3\_world.world. If the save dialog does not appear, try minimizing the Gazebo window to access it.
* Create a custom launch file to initialize your custom Gazebo world and the autonomous navigation system.
* Your launch file should start the Gazebo simulation with custom\_turtlebot3\_world.world.
* Ensure the launch file initializes the TurtleBot3 robot model and any necessary ROS2 nodes for simulation.
* Incorporate the execution of your autonomous navigation program within the launch process.
* Create a new launch file, e.g., custom\_world.launch.py, in the launch directory of your ROS2 package.
* After completing your simulation environment, custom world, navigation program, and launch file:
  + **Build** Your ROS2 Workspace:
  + Navigate to your workspace directory and run colcon build to compile your packages.
  + **Source** Your ROS2 Workspace e.g. source ~/.bashrc

*Tips: By installing the* [*colcon-clean*](https://github.com/colcon/colcon-clean) *extension, you can clean package workspaces*

sudo apt install python3-colcon-clean

colcon clean workspace

1. **Task 2: Autonomous Navigation Program**

Develop a Python program using ROS2 and OpenCV that enables the TurtleBot3 Waffle Pi to autonomously navigate the custom world without colliding with obstacles or walls.

**Requirements:**

* Your program should subscribe to the TurtleBot3's camera topic to receive images from its environment.
* Implement image processing techniques using OpenCV to detect obstacles and walls. Consider using color detection to identify red cubes.
* Based on the camera input, make decisions to steer the TurtleBot3 away from obstacles and walls. Your robot should aim to travel as far as possible without collisions.
* **The navigation logic should be purely based on the analysis of the camera feed (don’t use lidar)**.
* This [sample code](https://drive.google.com/file/d/1qIaWBkvMuD2utBdbAWLNbeii9Ho63QOk/view?usp=sharing) snippets for subscribing to the camera topic and controlling the TurtleBot3. Your task is to integrate obstacle detection and avoidance logic into this framework.

1. **Reporting (5%):**

* Write a 2-3 pages report in Arial, 11-point font, single-spaced, detailing your implementation and results for tasks 3 and 4.
* Describes your approach to creating the custom world and implementing the autonomous navigation.
* Discusses any challenges you encountered and how you overcame them.
* Explains the image processing techniques used for obstacle detection.
* **Video Demonstration**: Record a 1-minute video showing your TurtleBot3 autonomously navigating the custom world. The video should clearly demonstrate the robot avoiding obstacles and walls. Upload the video to a private YouTube link and include the link in your submission.

**Submission Guidelines:**

1. Submit your source code files, including the ROS2 package containing your autonomous navigation program (python) and **any modified or new scripts for the custom Gazebo world**.
2. Submit your report as a **PDF** file, ensure that all figures, and tables are properly labeled.
3. Your assignment should be self-contained, meaning a person should be able to understand your process and results just by reading your report and going through your code.

**Evaluation Criteria:**

1. Custom World: Correctness and complexity of the custom Gazebo world.
2. Autonomous Navigation: Effectiveness and efficiency of the autonomous navigation algorithm.
3. Obstacle Avoidance: Ability of the program to detect and avoid obstacles solely based on camera input.
4. Documentation: Clarity and completeness of the report and code comments.
5. Video Demonstration: Quality of the video in demonstrating the project's objectives.

**Note on Group Contributions and Grading:**

If any group member believes that another member of their group deserves a lower grade due to their contribution level, they are encouraged to address this concern. To formalize this, the group can include an additional section in their report detailing the situation and the proposed grade adjustment, with the consent of all group members. It's essential that all group members agree and provide their consent for any proposed grade changes.

It's always best to communicate openly within your group and seek collaborative solutions. However, if discrepancies in contributions are significant and consensus is achieved, this mechanism ensures fairness in grading.

**Best of luck! Remember, the process and learning are as important as the final results.**