Sensors & Control Mid-review Report 13224466

Sensors and Control for Mechatronic Systems (University of Technology Sydney)

Sensors & Control

Mid-review Report

Group 4

Table Of Contents

Table of Figures

1 Introduction/ Project Goal

For 41014 Sensors and Control, we have had the opportunity to work on our choice of a project which involves sensing and controlling. In our case, our project involves using the Fetch robot to follow the path of another robot (Turtlebot Waffle) via path planning algorithms. This is to be carried out through sensing and analysing the data obtained from the RGB and laser scanner onboard the Fetch robot to control and move the Fetch robot accordingly. A special QR code has been positioned on the guider robot which the Fetch robot will scan to extract the pose. We have been using C++, Linux and ROS skills to carry out the project. This project is conducted in a group of three where each team member has been assigned a specific task for the completion of the project.

My assigned role was to create the ROS launch file for the Gazebo environment to load in the Fetch robot, Turtlebot and associated models. I also had contributory and reviewer roles in other tasks.

2 Team member contribution summary

As the project was carried out as a group, each team member is responsible for specific tasks for the parts of the project completed so far. A breakdown can be seen below.

Tasks	**Student Roles** (**P** = primary **C** = contributor **R** = reviewer)		
Creating virtual environment	R	C	P
Image processing	C	P	R
Develop pure pursuit algorithm	P	R	C
Utilising real Fetch robot	C	P	R

Figure 1: Project Task Breakdown

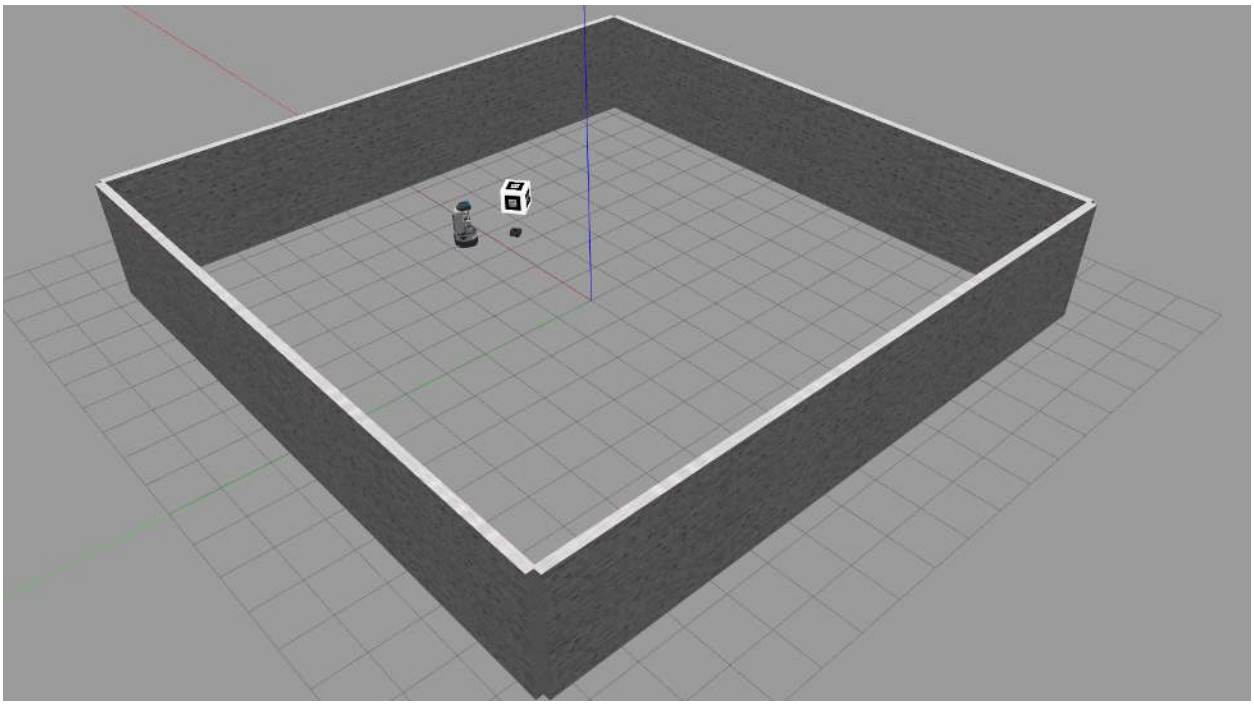
3 Personal project progress

3.1 Using Roslaunch to start Gazebo, world files and URDF models

Creating a ROS launch file for the Gazebo was the first task involved in the project as it sets up the Fetch and Turtlebot in an environment to interact within. It also allows for the input of obstacles and models to be involved with the Fetch robot.

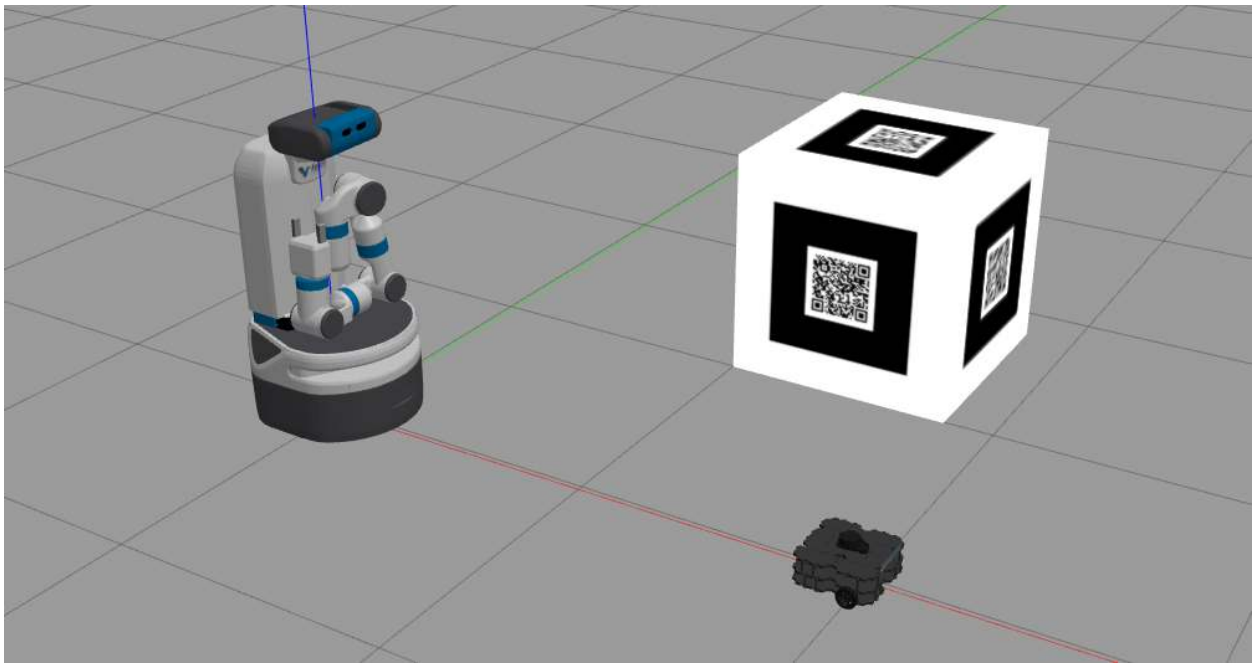
3.1.1 Creating a world file

A world file was created of which I included various models and elements. This includes a ground plane, a light source (sun) and eight walls (two on each side). This is most likely not what the final environment will look like but the intention was to not have a complicated environment to avoid long loading times when booting up the launch file. The environment can be seen below.



3.1.2 Creating a launch file

The purpose of the launch file is to spawn the relevant robots into the Gazebo environment, in this case, we have spawned the Fetch robot and the Turtlebot which have been obtained from installing and cloning the relevant packages into the catkin workspace. The file is also used to launch the robots into the previously created environment contained in the world file along with default parameters. The spawned robots along with the visual marker can be seen below.

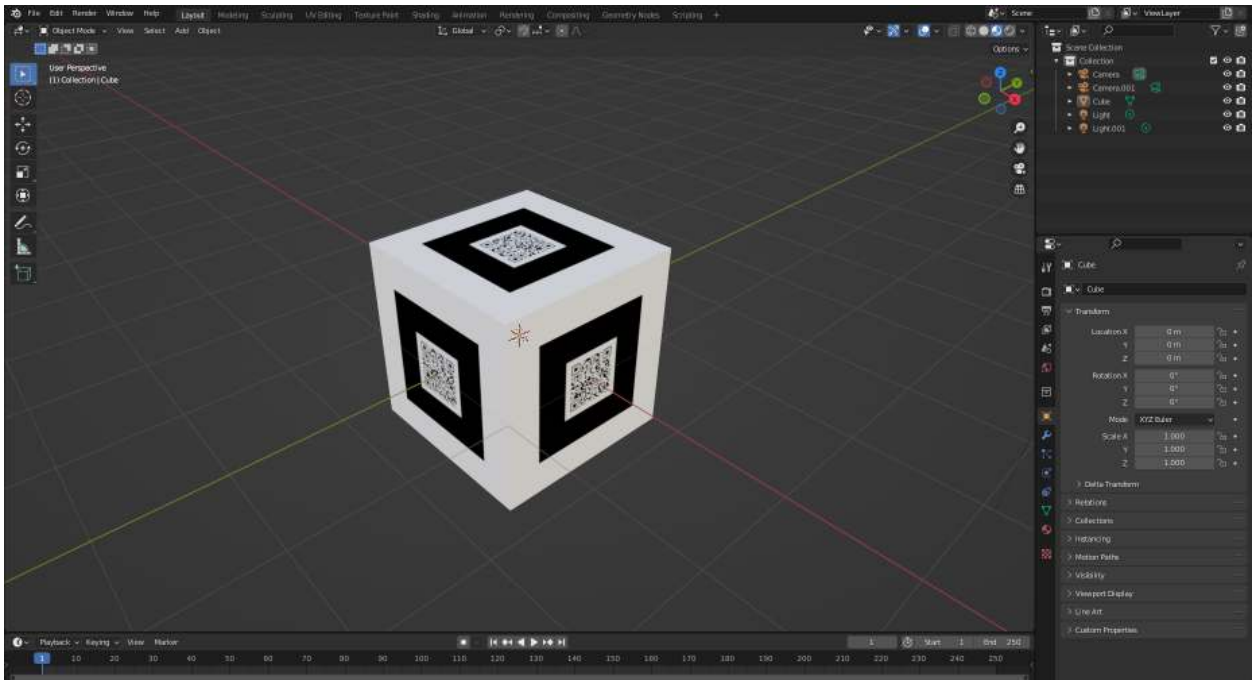


3.1.3 Creating QR code cube

It can be seen that the Turtlebot has a QR code cube hovering above it. This is for the Fetch robot to scan and track via the vision_vsp package. This computes the pose of the QR code relative to the Fetch robot which is about the same as the pose of the Turtlebot relative to the Fetch robot as the QR code is directly above the Turtlebot.

A cube was used to account for when the guider robot turns. Therefore, even when the guider robot is at an angle relative to the Fetch robot, there will still be a clear QR code to track and detect using another side of the cube.

The QR code was obtained from the vision_vsp ROS Wiki website as a png file. It was imported into Blender and placed on all four sides of a cube as seen below. The cube was then exported as a .dae file from Blender.



3.1.4 Modifying Turtlebot to hold QR code visual marker

To attach the QR code cube onto the Turtlebot, the (Unified Robot Description Format) URDF file of the Turtlebot was edited. The QR code was attached as a joint and a link to be scaled and positioned 1.05 metres above the base of Turtlebot.

3.2 Basic path following

Once both robots were set up in the environment, along with the QR code visual marker. A basic following algorithm was implemented using a ROS node. In the node, a subscription was made to vision_vsp (/visp_auto_tracker/object_position) and a publisher was made to the Fetch movement topic (/cmd_vel). From this, if the QR code is seen on the left of Fetch, the Fetch will rotate left. If the QR code is seen on the right, the Fetch will turn right and if the QR code is seen straight ahead (within 20 degrees field of view), the Fetch will proceed straight. The CMake Lists.txt file was also edited to support the Rosnode. This was only a basic path following algorithm which was initially used. My teammates have improved on this by implementing a pure pursuit algorithm.

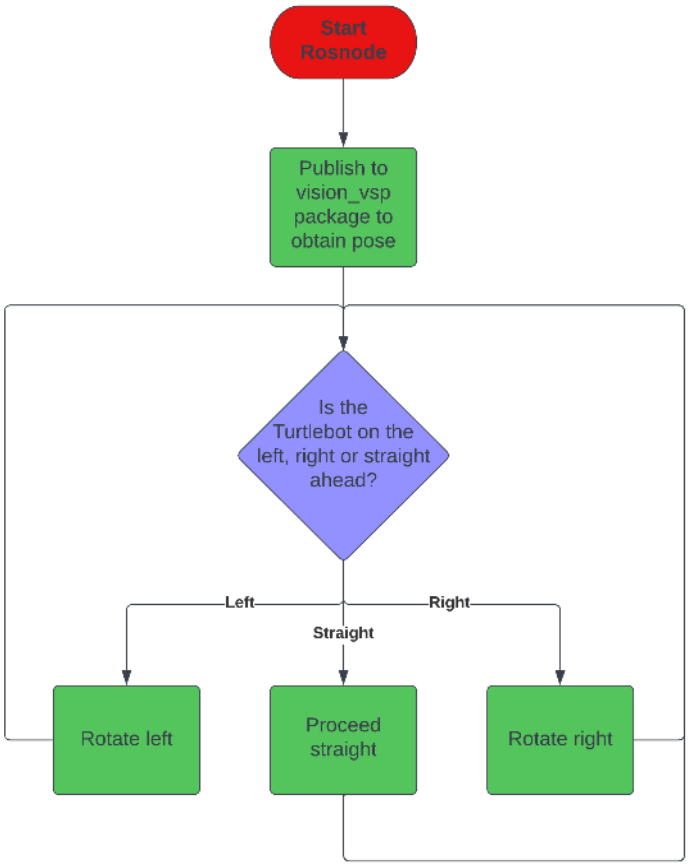
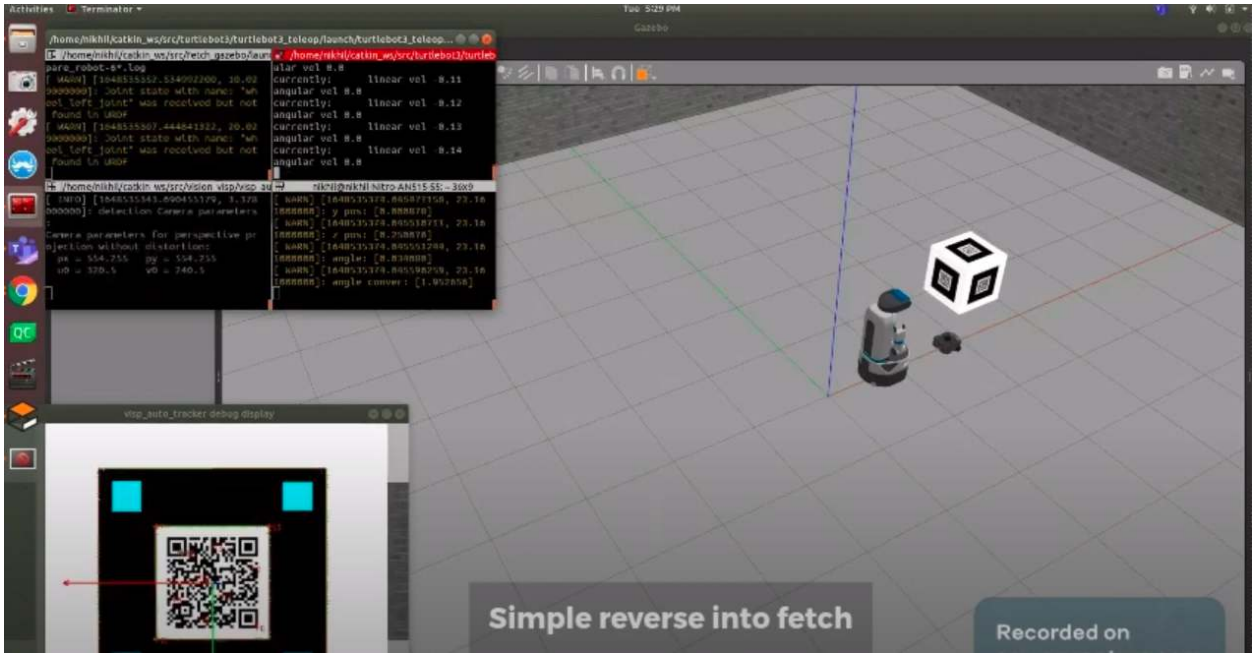


Figure 2: Path following process flowchart

3.3 Simulation demonstration and video

Once the Gazebo environment and basic path planning were all set up, we simply tested it by running the ROS path following node and controlling the Turtlebot using the keyboard controls. We then presented a demonstration video of the simulation to our supervisor and subject coordinator, Dr Liang Zhao. To him, we demonstrated simple path following, turning and stopping. However, once asked about reversing the guider robot by our supervisor, we realised the guider robot would crash into the Fetch robot.

We resolved this in our code and created a quick demonstration video of our Fetch project before we used the real Fetch robot. This demo was before pure pursuit was implemented and only demonstrates a simple path following algorithm. The link to the video is: https://youtu.be/flMWIfIf0FI



3.4 Using the real Fetch robot

During the STUVAC week of the semester, we had the opportunity to implement our path planning algorithm into a real Fetch robot in the Robotics Intelligence laboratory. We had the real Fetch robot follow a QR code that was attached to a post and moved around by a team member as using a Turtlebot (like in the simulation) would have been too complicated. We had demonstrated the Fetch robot following the QR code when going straight, turning, reversing against it as well as following it around obstacles.

The full video of our group using the real Fetch robot for this project has also been uploaded to Youtube and can be accessed through this link: https://youtu.be/eHr32ojcy4E

4 Issues during personal progress

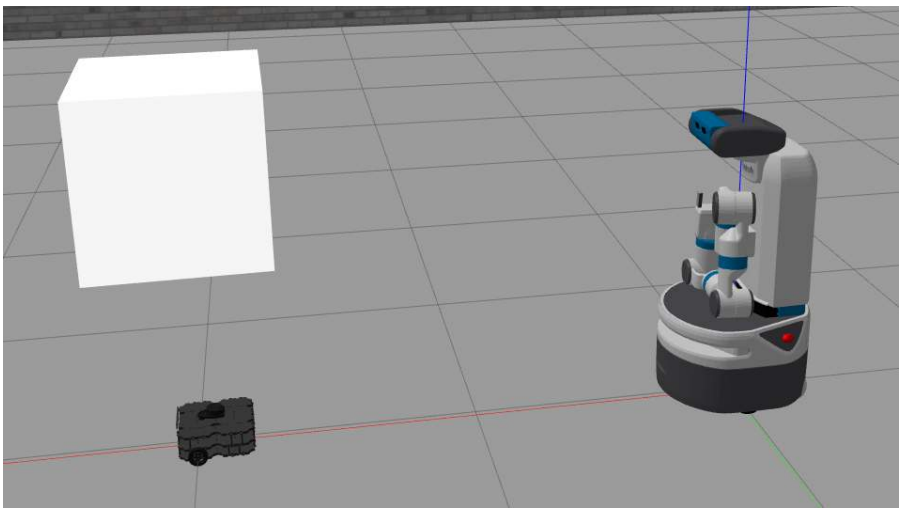
During this process, I expected to have problems and get stuck along the way. Most of these issues have been resolved through research and simply attempting possible solutions.

4.1 Inserting Turtlebot and Fetch together in Gazebo

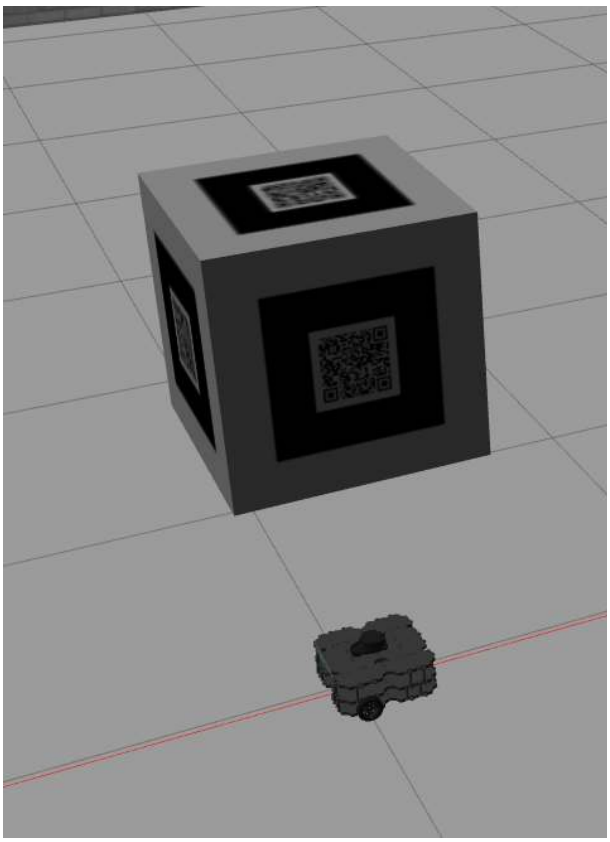
I had issues where the Turtlebot would not appear in the simulation and only the Fetch would. Only after referring to “Tutorial: Using Roslaunch to start Gazebo, world files and URDF models”, I realised that any URDF to be spawned in Gazebo should be in XML format. The Turtlebot was in XACRO format and so had to be converted before being spawned.

4.2 QR code cube not showing or too dark

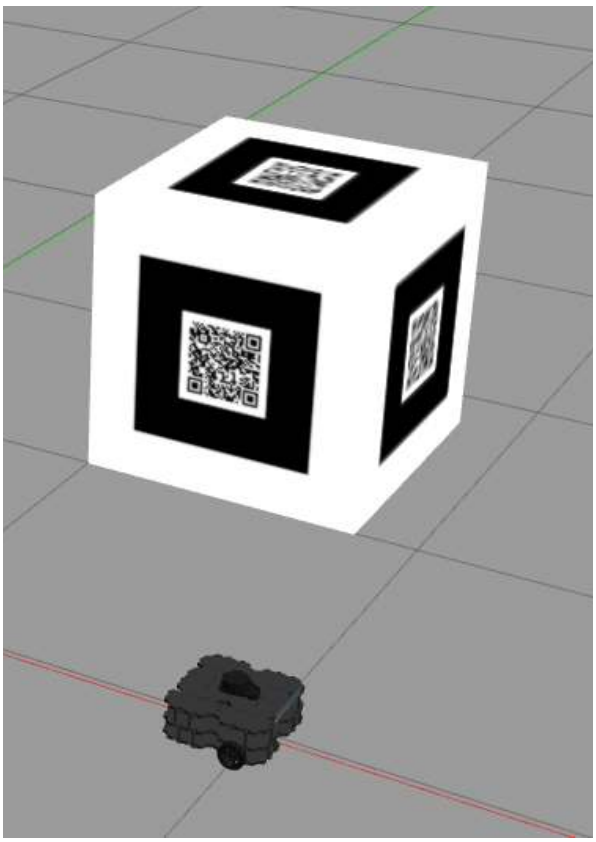
Another issue I encountered was the QR code cube in Gazebo either appearing too dark or just appearing as a white cube. This was a major problem for the project as having a clear visual marker was essential for the Fetch robot to calculate the pose and initiate the path following. Therefore, I could not proceed with the project until this was rectified. After online research, I found out that the cube was appearing white as I exported/ created it as an STL file that does not have colour. Instead, I exported it as a DAE file. I also realised I had to include the original image (png file), in which the cube was created, included in the folder.



After this, the QR code cube appeared in the Gazebo simulation, however, it seemed to be too dark to the point where it could not be detected. After conducting online research, I have deduced that the amount of light emitted from a DAE model in Gazebo can be edited from its source file. After simply increasing the light emission of the QR code cube from 0 to 0.8, the cube appeared to be much brighter and stood out enough to be easily detected from the Fetch's RGB camera.



BEFORE

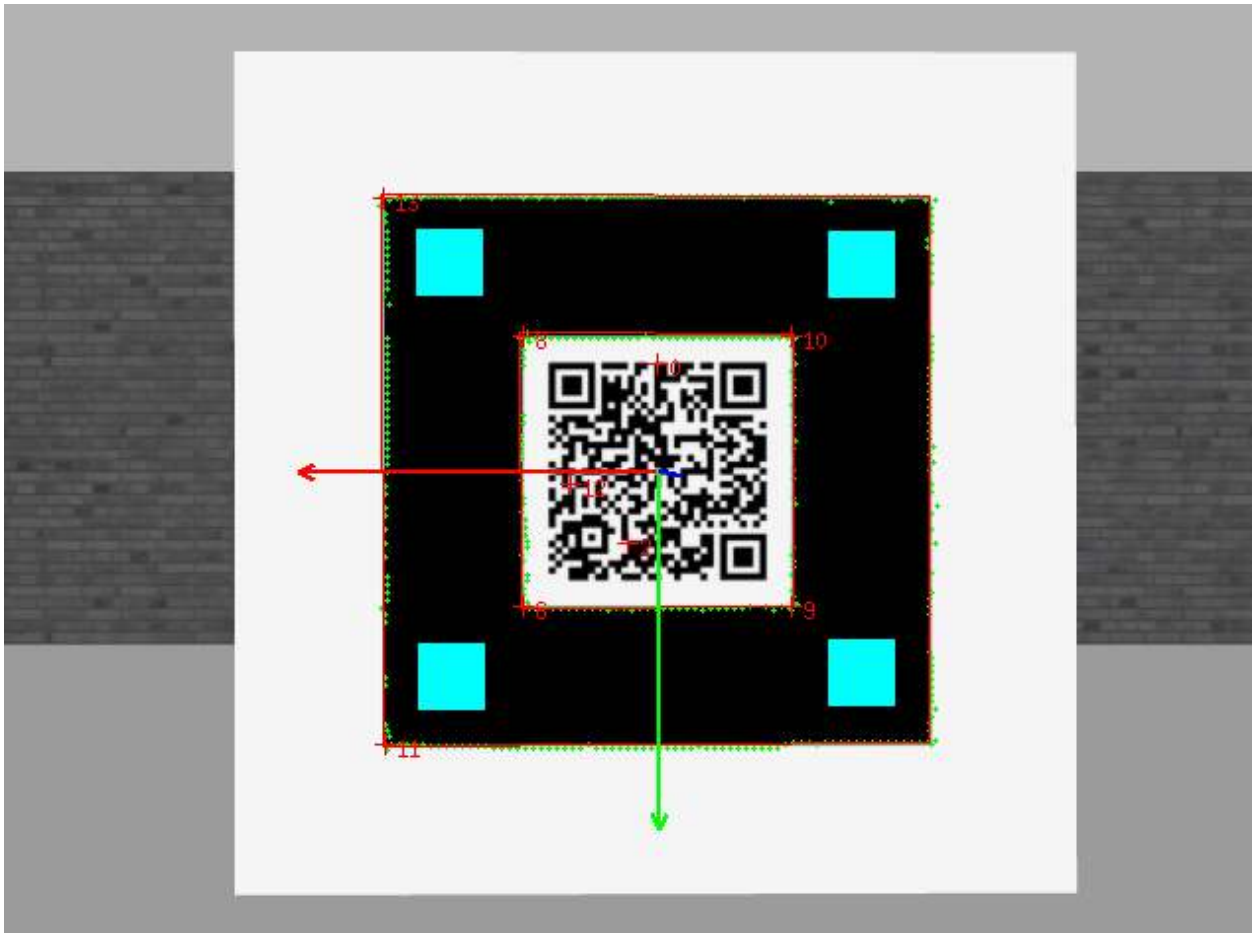


AFTER

4.3 QR code not being detected/ pose not being calculated

There were certain instances where the QR code was not being detected from the Fetch camera. This was an issue as the QR code needs to be detected to calculate the pose (position and orientation) of the Fetch robot to the QR code. After some experimenting, I realised the package being used for QR code detection requires a large black border around the QR code itself.

Furthermore, there is an ideal size and scale for the QR code to be readable by the Fetch robot. I continued to test various sizes until I achieved substantial results with the path following.



4.4 Simultaneously controlling Fetch and Turtlebot

The main method to control the Turtlebot (guider robot) was using the **turtlebot3_teleop** command. However, I noticed that when I first tried to use this, I ended up controlling both the Fetch robot and the Turtlebot together with the keyboard controls instead of just the Turtlebot. After looking into the source files for the Fetch robot and Turtlebot, I realised that they both share the same topic for movement, **cmd_vel** and that this was the reason. To simply resolve this, I changed the name of the movement topic for the Fetch robot to a different name (cmd_vel1) so that the Fetch robot would not be affected when using the Turtlebot movement.

5 Improvements & plans

As a team, we have so far been able to carry out path following for the Fetch robot and we have also implemented our code into the real Fetch robot. However, there were still certain aspects of the project that can either be improved or newly implemented.

5.1 Obstacle detection/ collision avoidance

As of now, the Fetch robot can follow the QR code as well as reverse against it if it comes too close. However, if an obstacle were to obstruct the path of the Fetch robot, it would simply crash into it as we have not yet implemented obstacle detection.

We plan to utilise the onboard base laser on the Fetch by subscribing to the topic, **base_scan.** This will allow us to detect any obstacles within 25 metres and a 220° field of view. From this, our plan is for the Fetch to stop before any obstacle in between the QR code and itself where it should only continue moving until the obstacle is removed from the path. Later on, if there is time, we also plan to utilise the **odom** topic which can help the Fetch robot go around the obstacles rather than just simply wait for them to be removed. The process can be seen below in the flowchart.

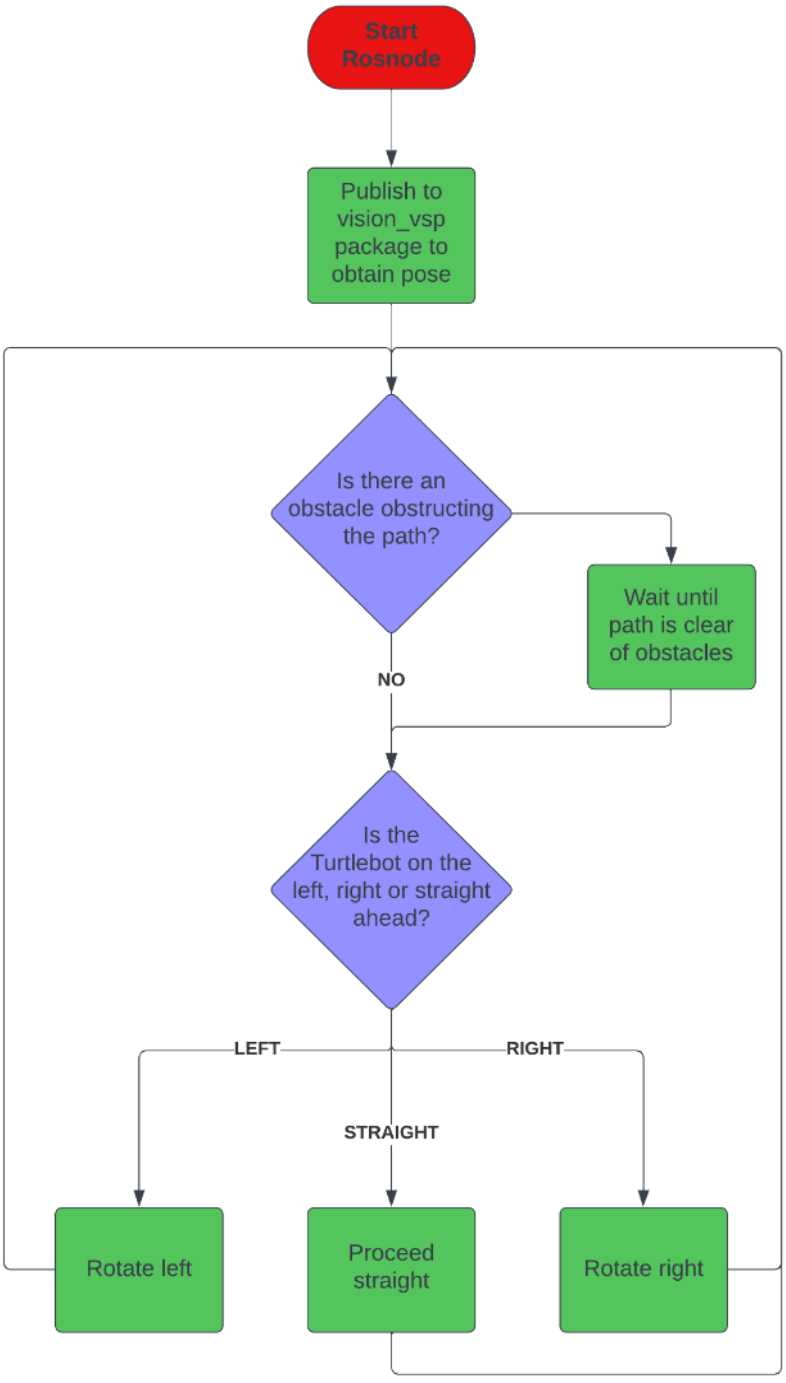


Figure 3: Path planning and obstacle detection process flowchart

5.2 Detailed Gazebo environment

Currently, our environment simply consists of a square wall compound along with the two robots. As it was found not too difficult to create a detailed environment, it was decided it would be left as a task to do later on until more important aspects were completed. Furthermore, initially having a detailed environment would lengthen to load up time for the simulation and would therefore not be ideal as we are constantly updating and loading the simulation during this time. We plan to incorporate a detailed office and corridor style environment as this is the target of the project. We also intend to include obstacles in the environment for the Fetch robot to avoid.

5.3 Playstation controller to control Turtlebot

At the moment, the Turtlebot (guider robot) is being controlled via keyboard controls. Although it is not a major goal for the project, we plan to use an external controller such as a Playstation controller to control the Turtlebot in the simulation. For the real Fetch, we simply placed a QR code onto a post which we move around by hand.

5.4 Utilise the Odom topic to track path following

From observing the available ROS topics for the Fetch robot, we have observed there is an odometry topic. We plan to attempt to use this topic to keep track of where the Fetch is in the global coordinate frame and relative to the QR code as it is path following. This would help in the case where the Fetch robot loses sight of the QR code as it could search where it last saw it to increase the chances of relocating.

5.5 Control linear and angular velocity simultaneously with real Fetch

When using the real Fetch robot in the Robotics Intelligence laboratory, we noticed that it does its translation and rotation independently of each other even though the pure pursuit algorithm outputs both linear and angular velocities simultaneously to the Fetch robot. After talking with the laboratory supervisors, we believe it is due to how the internal drivers of the Fetch robot

operate and that a different Fetch robot may have to be used. However, we plan to investigate further into this and eventually resolve this to ensure smoother motion with the real Fetch robot.

5.6 Use a hexagon instead of a cube

For the visual marker which the Fetch sees is currently a cube. This is so when the guider robot turns, another side of the cube holding the QR code can be seen by the Fetch. This allows for a higher chance of a QR code to be seen as the Turtlebot turns. As an improvement, a hexagon shape could potentially be used to further increase the chance of any single QR code being seen by the Fetch robot.

6 Conclusion

To conclude, our group have been able to develop a path following algorithm for the Fetch robot to follow the guider robot while maintaining a set distance for the 4014 Sensors and Control group project. We have been able to demonstrate the project both in simulation and through the real Fetch robot. However, we still have further tasks to work on for the project for improvement.

7 References

Wiki.ros.org, (2022), visp_auto_tracker, ROS Wiki, <http://wiki.ros.org/visp_auto_tracker>. [Accessed 18 April 2022].

Gazebosim.org, (2022), Gazebo: Tutorial: Using Roslaunch to start Gazebo, world files and URDF models, Gazebo Simulation <http://gazebosim.org/tutorials?tut=ros_roslaunch>. [Accessed 18 April 2022].

Answers.gazebosim.org, (2022), imported mesh with texture too dark in Gazebo, Gazebo: Q&A Forum. <https://answers.gazebosim.org//question/8475/imported-mesh-dae-with-texture-too-dark-in-gazebo/>. [Accessed 18 April 2022].