**Issues listed in the real environment**

**Colour detection**

**Lab background issue**: The background also has the same colour as the object. For example, in the lab, there is a red locker and reboard. When the robot detects the red locker as a red colour and then stop.

**Solution**: Cover the red locker and red board with the other color. Professionally, make a classifier to identify the color card. Then, the other object will not be detected. On the other hand, try to capture the color card hsv with close range so the camera will only detect the red card. For example, in my current program, the color detection of the red can detect all the red (pink, blood red etc). We can try to set the hsv range value to detect blood red only.

**Color noise issue**: the camera processes some images, and it appears color noise which refers to some pixels will appear red, blue and green and only the camera can detect it. When the programming runs it, my program counts the color noise as a color card.

**Solution**: Image denoising (OpenCV). It may decrease the Image noise, accurately finding the red, blue and green card. In assumption, we can make an if condition to filter out small pixels. However, it will take a lot of time to check the pixel size on the color card as the distance between card and turtlebot will affect the pixel size.

**Red line Follow**

● **The ratio at simulator and real environment**:

The ratio between the actual robot and objects has errors in the real life when comparing the simulated environment to the real environment. From that, the robot could smoothly run in the simulated environment but in the real environment the ratio has changed, coding has to be changed as well to make it run.

**Solution:**

- Change the proportional controller is implemented which responsible of linear scaling of an error to drive the control output

In the simulator:

err **=** cx **-** w**/**2

self**.**twist**.**angular**.**z **=** **-**float(err) **/** 100

In the real environment:

err **=** cx **-** w**/**2

self**.**twist**.**angular**.**z **=** **-**float(err) **/** 700

- Re-scale the ratio of all objects in the real environment and set up the map in the simulator.

● **Subjective factors**: the implementation time for this project is quite short (13 weeks) when compared with others.

**Solution**: The project should be longer such as 2 study periods or contains double units. If the period could be in 2 study periods, we could have more time to research and more time to work together. Also, if the project has double units, we will need to do 1 or 2 other subjects. From that we could have more time and more focus for the project.

● **Objective factors**: our backgrounds are general information technology, so we have not been prepared strong enough for an engineering foundation to work with the robot in the real environment.

**Solution**: Show the demonstration in the simulator which does not require much engineering skills.

**Obstacle avoidance**

**Basic background and connection issue:** If we want a robot to complete these tasks in a real environment, we need a very strong engineering foundation. The basic issues I met were mainly connection issues between the laptop and the robot, including network configuration failure, very limited hardware knowledge. As an IT student, with the lack of engineering foundation I can only implement the robot functionalities in the simulator using Python rather than in real environment.

**Solution:** I will take time during the holidays to learn the engineering knowledge related to this project. Including solving the connection issues between laptop and robot connection and being familiar with robot hardware.

**On-site operation issues:** The robot would only spin around in place. For the first time in the lab, I tried to adjust the linear velocity, angular velocity and detection distance values, but the robot still only rotated in place. When I increased the linear velocity and angular velocity, the robot rotated in a large circle on the spot; when I decreased the linear velocity and angular velocity, the robot rotated in a small circle on the spot. The second time I tried to swap the values of the X axis and Z axis, but it still did not work.

**Solution:**

I need to check the real published node from the turtlebot of the laser data, and then correspondingly modified the subscriber code. In addition, continue to adjust parameters.

**Different obstacle sizes issues:** In the simulator, the robot was very small, and the obstacle was very large. But the size of the real turtlebot in lab is bigger than the obstacle. Therefore, the avoidance calculation function applied to the simulator is also undesirable in the real environment.

**Solution**:

I can update calculation function of linear velocity and angular velocity then test parameters repeatedly, even though all the calculations and parameters in the entire project may have to be changed, which could introduce some bugs and take more time.