CISC 3060 CLASS PROJECT: Color Task

1.0 Overview

Your class project is to construct a ROS program that will control **either** a) a Turtlebot3 simulated robot in Gazebo, or b) an actual Turtlebot3 in the robotics lab so that it uses visual navigation and obstacle avoidance to search for and go to each of a number of colored targets in turn while avoiding obstacles. This is called the *color task.* The simulated and real robot projects **differ** slightly so pick your choice carefully:

- The simulated robot color task will require **four colors** to be located in sequence and there are **four obstacles** to avoid.
- The actual robot color task will require **two colors** to be located in sequence and there is **one obstacle** to avoid.

Apart from this, the two different color tasks have exactly the same requirements. Do not be fooled into thinking the real robot task is easier because it has few colors and obstacles. It is very likely harder and certainly will require the work to be done in the robotics lab rather than on your virtual machine.

You will carry this project out in teams of 2, with each member responsible for **different** challenges and software that must work together and fit into the single team color task project. You will be graded on your individual contribution to the project as well as how well the team project was done overall.

In the rest of this document, the simulated color task will be used to explain what is needed. Anything required of the simulated robot color task is also required of the actual color task except where noted.

2.0 The Color Task

The color task Gazebo world is shown in Figure 1. A small section of each of the walls of the enclosed.world Gazebo world file has been replaced with differently color objects (black, white, yellow, orange). Four large blue columns have been placed in the center of the enclosure. The robot will start in the center of the four blue columns, and then navigate using visual sensing to each of the orange, white, yellow and black targets in that order. The robot cannot know the locations of the targets and must search for each of them visually.

In the actual robot color task, two colored pages will be affixed to the walls of a hexagonal enclosure approx. 1m on a side. in the lab. Two colored pages (pink and blue) will be affixed to two arbitrary walls. There will be a single central obstacle. The task is otherwise the same.

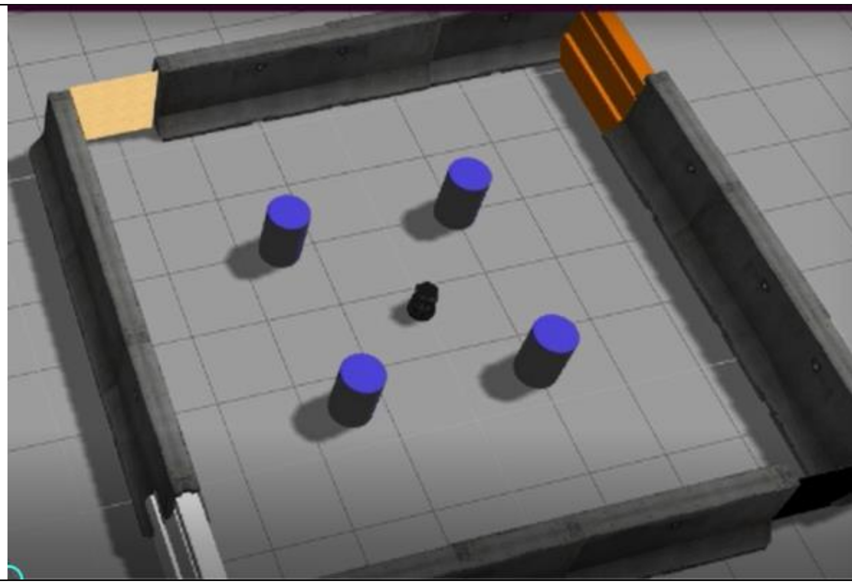


Figure 1: The Simulated Color Task World

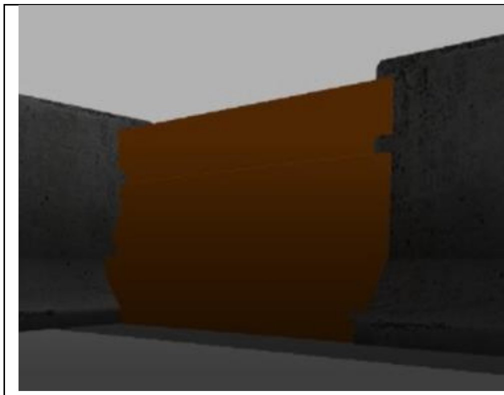
Once the robot is close to the first target, it should calculate its position and the time that has elapsed since it started. It should write this information into the image of the target and store it as a file called goal0.jpg. It should store goal1.jpg after it finds the second target and so forth. The objective is to make the run *as quickly as possible*, but without any knowledge of the locations of the targets, and without hitting anything along the way.

Part of your final presentation will be to run the course from a random start position that you will be given. The fastest team wins! Simulated and actual robot runs will be judged separately. Winning qualifies you for a certificate but your class grade will only reflect whether the robot was able to complete the task or not.

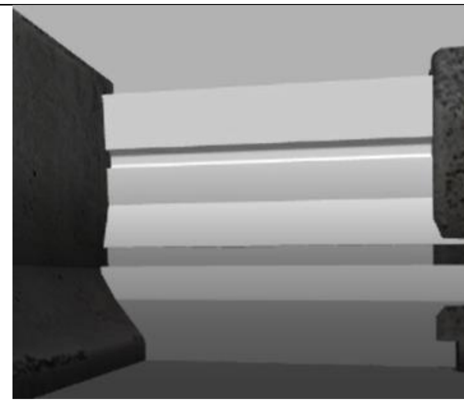
3.0 Color targets

There are 4 color targets in the simulated tasks and 2 in the actual task. You can identify the targets using the cv2.inRange function. The upper and lower color ranges for the simulated targets are shown in Figure 2. You can certainly adjust these values if you think you can get better performance. (Request the values if you are doing the actual robot task).

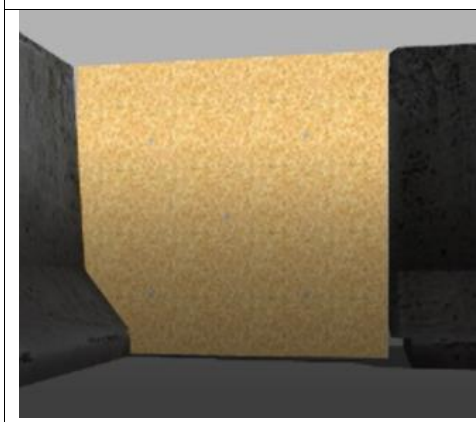
You should use the cv2.moments function to extract the area and target center. You can use the measurement of area as an estimate of the distance from the target. An area of 50 is probably close enough to the target (and 80 is probably too close) for all of the targets. Again, you can adjust this value if you think you can get better performance. However, the robot does have to be close to the target (informally) before it can consider the target reached. It cannot just see the target from the other side of the enclosure and consider that as enough.



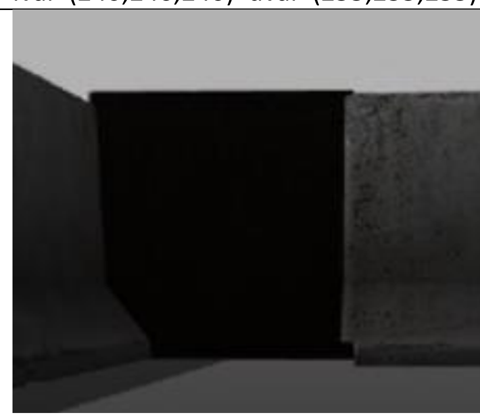
lval=(0,30,75)   uval=(5,50,89)



lval=(240,240,240) uval=(255,255,255)



lval=(0,80,80)   uval=(10,110,110)



lval=(0,0,0)   uval=(20,20,20)

Figure 2: Color targets

4.0 Recording target achievement

After the robot has moved to within a close distance of the target (e.g., area=50 is a good measure), you must record some proof of reaching this goal. You should save the current image to a file. The file should be called goal0.jpg for the first target, goal1.jpg for the second, etc. You also need to add some information to the image. You should write:

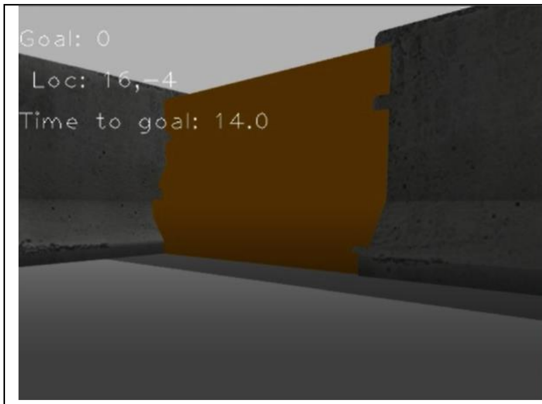
a. the goal number 0 through 3;
b. the x and y location at which the robot stopped;
c. the time that has elapsed since the task started.

Figure 3 shows an example for each of the four color targets. You can use whatever color, font and scale you want to write on the image as long as you write this information (a through c above). White was used in Fig. 3, but it does not show up that well in my opinion.

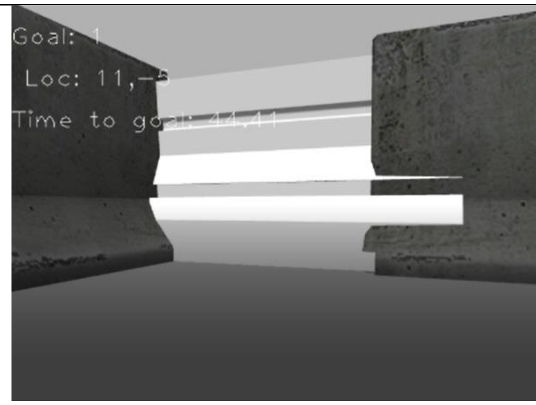
5.0 Structure of the program

You must structure your program ***as two ROS nodes***. The first node accepts a Twist() message on a topic you get to name (e.g., safe_cmd_vel) and will pass this to the robot **if there are no obstacles detected** by laser on the left hand side or the right hand side. If there are obstacles, then the angular velocity is *not* passed to the robot, instead -0.2 is passed if the obstacle is on the left and 0.2 if the obstacle is on the right. Furthermore, if there is a left or a right obstacle, the linear velocity passed to the robot is *one half* *of the linear velocity passed as a command* on the input topic. As you can see the purpose of this node is to separate any obstacle avoidance concerns from the color tracking concerns. Let's call this the OANode.

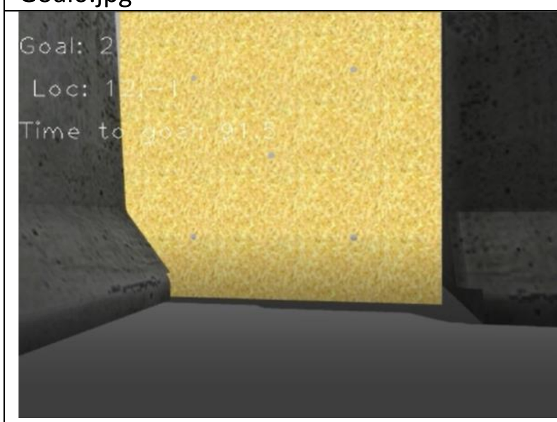
You may want to modify this behavior so that obstacle avoidance will only be triggered if a nonzero linear velocity was sent to the input topic. If the linear velocity is zero, then the robot may just be spinning, and no obstacle avoidance is needed.



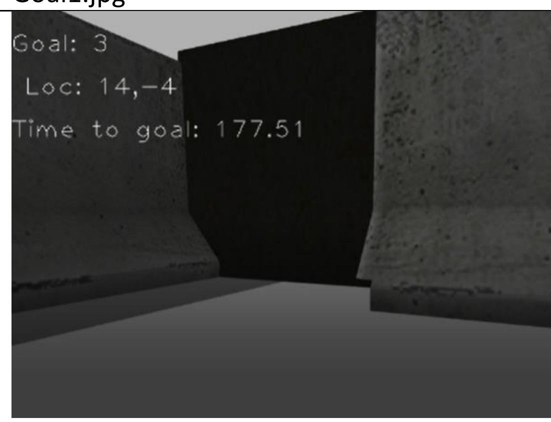
Goal0.jpg



Goal1.jpg



Goal2.jpg



Goal 3.jpg

Figure 3: Images saved for each target

The second ROS node is responsible for (1) searching for the color target and (2) when it is found, generating a linear and angular velocity to bring the robot close to the target. Let's call this the CTrackNode. The output Twist() message from the CTrackNodeNode should be written to the input topic of the OANode, which will then override the velocity on occasion to steer the robot away from obstacles.

6.0 Deliverables

Each student will need to write an individual report that includes all the details of the design, implementation and testing of the node they worked on, and then all the testing and results for the joint project with both nodes. You will work in teams to build and test the software. All the rules that related to lab reports related to this final document. Write it as well and as completely as you can. Your report must be all your own work and unique to you. Your final system will be a group effort but each member will submit their code with their report.

Your team will also need to do a 10-minute presentation (on the final exam date) on your design explaining how your program works. You should USE as much ROS and robotics algorithms as you can from this class!! Don't reinvent the wheel. Each team member must participate in the presentation.

Finally, each team will be given a random initial location for the robot (close to the center of the enclosure) and must run their program and deliver their four (or 2) images and times. The performance of all the team systems will be compared and a winner announced!