****

**MEEM 4707: Autonomous system**

**Spring, 2024**

**Lab - 6**

**By**

**Colton Kreischer**

# **Problem 1.** Use “perception\_camera.py” to do trials and find the mask range.

1. Tune “lower\_range” AND “upper\_range” in the code (the HSV range) to mask the image for the red cylinder.  You may use tools available online to determine the approximate HSV value of a color.

I found the values by printing all unique HSV tuples from the image out, then reasoning what colors each set of similar values would be generating. I found that the red hue did not deviate much from 0 (range 0-10). I found that the minimum saturation and vividness values were all both above 10, while an upper limit is not needed for this specific environment (no reflection/dissipation/competing light sources/colored light sources/camera artifacts; therefore, only the red hue needs to be identified and the grays filtered out by low saturation).

Final ranges are shown below.

A computer screen shot of a program

Description automatically generated

1. By using your range, provide mask images in the report.   
   Note: The cylinders should have ample lights.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# **Problem 2.** Assume that the robot orientation is initially such that your mask image for the red cylinder looks like Figure (1).  This mask image is basically a 2d matrix that has only “0” (dark zone) and “255” (bright zone). Assume that the size of the image is 648 x 648.

1. What would be your approach to finding the distance of the centerline of the bright zone (shown in red) from the centerline of the image (shown in yellow)?

To find the centerline of the cylinder, the centroid of the mask can be calculated by taking the average X and Y coordinates of all pixels that pass through the mask. The X coordinate can be used to generate the red line, while the yellow line is presumably in the center of the image (X=324).

If a physical distance is needed from this data source alone, then the actual area of the cylinder that is seen (or its radius and height) must be known. If it is, then the distance can be found by using the ratio between areas and the inverse square law to approximate its distance away. The distance between the centerlines, assuming they are equal distances from the camera, can then be found using trig (where an arc length about the camera separates them).

1. Using “if-else” conditions, write conditions and speed commands to change the robot’s orientation until the centerline of the cylinder comes close to the centerline of the image as per your requirement, like in Figure (2). (You don’t need to write the entire code, write the if-else conditions and speed commands to share your approach)

A computer screen shot of a program code

Description automatically generated

**While a PID loop for px\_error would be more appropriate, this should still work using only if/elif/else.**

****

**Problem 3.** Read the “perception\_camera.py" and summarize your understanding.

1. What are the variables and command lines in the code define the matrix size of the mask image?

A computer code with colorful text

Description automatically generated

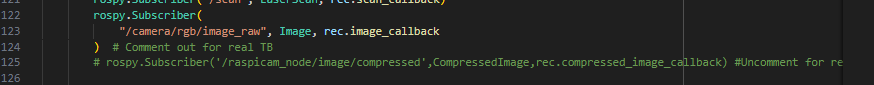


The mask’s matrix dimensions will match those of hsv\_image (height, width). The only difference is that the third dimension, the 3 channels, will instead be replaced by one channel; this new channel is only 0/1, and represents failing/passing the mask’s conditions instead of RGB or HSV color data.

1. Point out the command lines in the code that should be commented out and uncommented for the real robot.

A computer screen with text

Description automatically generated



1. Explain how to prevent the mask image from popping up every iteration.

This can be prevented by simply commenting out the following code:

A computer screen shot of a computer code

Description automatically generated