Intelligent Robotics MSc Summer Technical Challenge

Mobile Robot Navigation in Python

Mark A Post (mark.post@york.ac.uk)

1. Aims and Objectives

This is the third step in the Intelligent Robotics MSc Summer Technical Challenge.

This challenge can be done from home, either by starting a PC or VM from a bootable Ubuntu image or by installing the needed software on your own computer. It is intended to set you up with a simulation and programming environment that you can use anywhere to complete robotics laboratory tasks.

2. Learning outcomes

By the end of this challenge, you should be able to:

- Create a robot simulation in V-REP and control a robot with Python
- Test algorithms on virtual robots in place of real ones in the laboratory

3. Software and hardware

- V-REP (CoppeliaSim) Software (https://www.coppeliarobotics.com/)
- Example simulation environment (v-rep-lab.ttt) and control scripts (v-rep-lab.py and v-rep-maze.py) provided in the Appendix.
- Python or iPython development environment (Spyder3 is recommended)

4. Pre-Lab Preparation

- This challenge is a look ahead at the topics on machine vision we will be covering in the labs of your Practical Robotics module, but we will keep it very simple using an example of finding the centroid of a color in an image. Use an internet search to look up suggestions of algorithms that can be used for finding the centroid of colour in an image. You will find that doing internet searches is a valuable way of learning new skills and finding examples when you are trying to solve a problem. If you don't know it, don't hesitate to look it up!

5. Identify the Maze using Computer Vision

Your robot in the “v-rep-lab” simulation environment has a camera that you can use in a Python script to guide your robot using machine vision algorithms. As you may not have studied machine vision algorithms before, we will start you off with a simple algorithm that we will study later in the Practical Robotics module: centroid detection implemented using the Open Computer Vision (OpenCV) library.

The v-rep-lab.py script already filters the image that the camera sees into red, green, and blue channels, which themselves are black-and-white image matrices named “red”, “green”, and “blue” in the script. You can just take the “green” matrix and use it to guide the robot down the green line by detecting the center of the monochrome “green” colour (which you view as a white shape in the appropriate OpenCV window). When the centroid of this colour moves to the left (in y-coordinate) of the vertical center line of the image, you should instruct the robot to turn left. If it moves to the right you should instruct the robot to turn right, and of course, the more this centroid moves to the left or right, the more turning speed should be applied. As the camera on the robot produces a 256*256 image, the centre line is at *y==127*.

An example of finding a centroid in OpenCV can be found (as you might in the pre-lab preparation) at:

https://learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/

With a slight modification of the code here, you can easily find the centroid of the green colour and then use the y-coordinate to determine which direction to turn. Try starting with the following Python code inserted right after where the red, green, and blue channels are filtered in the v-rep-lab.py script:

```
#Find the distance from the center line to the centroid
# convert the grayscale image to binary image
ret,thresh = cv2.threshold(green,127,255,0)
# calculate moments of binary image
M = cv2.moments(thresh)
# calculate x,y coordinate of center
cX = int(M["m10"] / M["m00"])
cY = int(M["m01"] / M["m00"])
# put text and highlight the center
cv2.circle(green, (cX, cY), 5, (127, 255, 127), -1)
cv2.putText(green, "centroid ("+str(cX)+","+str(cY)+")",
      (cX - 25, cY - 25),cv2.FONT_HERSHEY_SIMPLEX, 0.5,
      (127, 255, 217), 2)
```

Remember to keep the indentation consistent in Python – tabs are used in the example scripts.

Your task is now to use the value of *cY* to guide your robot toward the green path. You will need to change the variables *leftMotorSpeed* and *rightMotorSpeed* to turn your robot and move it forward. The further *cY* is from the value 127, the faster you should turn, and you should try to move the robot forward slowly at all times. Experiment with different algorithms and see what works well to guide the robot toward the green line.

6. Navigate Through the Maze

Using the examples and code that you have developed until now, the final challenge is to make a program that will guide the robot all the way from the start (blue) square in the maze to the end (red) square in the maze. You are encouraged to be creative in your solution and experiment with different robot control strategies. You can make use of the other channels of colour that are filtered in the python script, the odometry measurement, and you can also modify the simulation if you wish. Bring your simulation to the first week of the Practical Robotics module and show off your robot navigation!