Control of Mobile Robotics

CDA4621

Fall 2022

Lab 4

Navigation with Camera

Total: 100 points

Due Date: 10-31-2022 by 11:59pm

The assignment is organized according to the following sections: (A) Requirements, (B) Objective, (C) Task Description, (D) Task Evaluation, and (E) Lab Submission.

A. Requirements

A.1 Physical Robot

Robot: "Robobulls-2018". **Programming**: Python
Basic Files: tof.py ("SamplePrograms.zip")

A.2 Robot Simulator

Simulator: Webots. **Robot**: "e-puck". **Programming**: Python
Basic Files: "Lab4_epuck.zip"

B. Objective

This lab will teach you about: (1) Camera, and (2) Object Detection.

B.1 Physical Robot

B.1.1 Camera

Read the supplementary material[1] on accessing the camera from Python. OpenCV is a popular library for real-time computer vision. It can be used for accessing the camera on the Raspberry Pi. The camera is initialized by creating a "VideoCapture" object. Frames can be retrieved at any time using the "read" function. These frames are in the "Mat" format, which allows the frame to be easily manipulated or displayed on screen with the "imshow" function.

B.1.2 Object Detection

Read the supplementary material[2] on using the Simple Blob Detector feature in OpenCV. One of OpenCV's features is Simple Blob Detector, which allows for easy detection of blobs in an image, i.e. groups of connected pixels of the same color. This feature is initialized with a set of "Params" that specify the allowable size, color, and shape of blobs. Then a "Mat" object is input into the detector, which is a raw grayscale or color image. The detector returns a list of blobs in the form of "KeyPoint" objects. These "KeyPoint" objects contain the X and Y center position of

---

[1] https://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/
https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/

[2] https://www.learnopencv.com/blob-detection-using-opencv-python-c/
https://docs.opencv.org/3.4.1/da/d97/tutorial_threshold_inRange.html
https://docs.opencv.org/3.4.1/d2/d29/classcv_1_1KeyPoint.html

each blob, along with its diameter. One major optimization that can be done for the blob detector is by masking out parts of the image that are not relevant for the detector. If a specific color needs to remain visible, this can be accomplished by first converting the "Mat" from RGB (red-green-blue) format to HSV (hue-saturation-value) format with the "cvtColor" function. Then "inRange" function can be applied to black-out any pixels in the "Mat" that are not the correct color. This masked frame is then fed into the detector.

B.1.3 Camera Threading

Read the supplementary material[3] on using threading to improve camera performance. Achieving low camera latency is important for proper object detection and tracking. The process of capturing a frame consumes a small amount of time, and when combined with a computationally intensive task like object detection leads to increased latency. Too much latency causes functionality like proportional control, to begin to malfunction, unless movement is slowed down significantly. You may need to implement camera threading if you are experiencing these issues.

B.2 Robot Simulator

B.2.1 Camera

The e-puck robot includes a camera[4], having an image size of 80 pixels x 80 pixels. For this lab, the robot needs to recognize a single yellow color set with the *recognitionColors* field. The resulting image will be displayed inside a red rectangle on the upper-left side of the 3D window.

B.2.2 Object Recognition

Recognition mode is enabled with the "*camera.recognitionEnable*()" function. The recognition node has a "*recognitionColors*" field where the colors to be recognized are specified, and the camera will only recognize objects of this colors. Use "*recognized_object_array = camera.getRecognitionObjects*()" to return an array of all the objects being recognized by the camera and "*recognized_object = camera.getRecognitionObjects*()[0]" to return only the first object in the array. The recognized object(s) will be surrounded by a red rectangle and if you hover the mouse over it, you can see the information of the object (id, position, orientation, size, etc.). Use "*recognized_object.get_position*()" and "*recognized_object.get_orientation*()" functions to get the position and orientation of the recognized object at its center and they are expressed relative to the camera and the units are in meters and radians, respectively.
Use "*recognized_object.get_size*()" function to return a value X and Y sizes in meters relative to camera.

C. Task Description

The lab consists of the following tasks leading:

- Task 1 – Motion To Goal ("Lab4_Task1.py")
- Task 2 – Bug Zero Algorithm ("Lab4_Task2.py")

C.2 Task 1 – Motion to Goal

The program should begin by facing the yellow cylinder (goal), and then continue by moving towards it, as shown in Figure 1. Motion must stop once the robot is 5 inches away from the cylinder. There are no obstacles between the robot and the goal. Robot may be moved at any time during execution. The robot should react accordingly by facing the goal again and then moving towards it. Note that the goal may not be visible at all if the robot is facing a different direction. You will need to account for this by rotating the robot until the goal is in view and continue moving. In case the robot is moved further away, the robot needs to keep moving towards the goal and stop again at 5 inches from it. You may apply a PID controller. The input

[3] https://www.pyimagesearch.com/2015/12/21/increasing-webcam-fps-with-python-and-opencv/
[4] https://cyberbotics.com/doc/reference/camera

function can be the distance and orientation of the yellow blob, and the output can be the angular velocity of the robot's wheels. Task should run for 30 sec. During evaluation, the TA will start the robot at different distances and orientations. Note that in the physical environment, the yellow cylinder is 2 feet tall and 4 inches in radius, covered in non-reflective bright paper, and robot should be able to detect cylinder from up to 8 feet in distance.

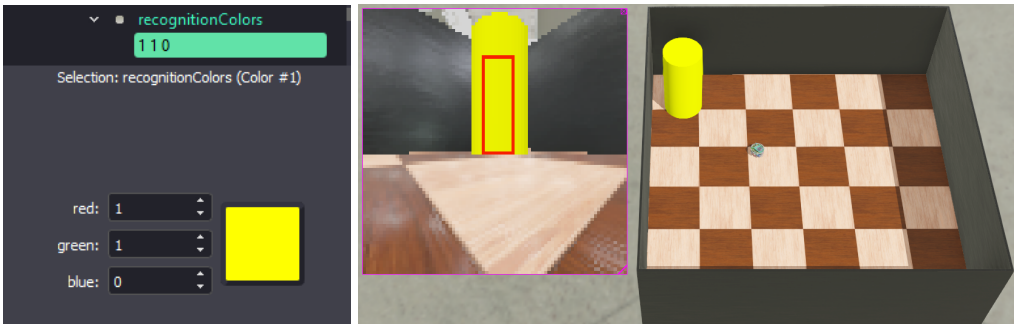

Figure 1. Camera color object recognition in robot simulator for motion to goal.

C.3 Task 2 –Bug Zero Algorithm

The robot should follow the Bug Zero algorithm, using the world shown in Figure 2. The goal is represented by the yellow-colored cylinder. The robot should follow walls no further than 5 inches away for the simulated robot and 8 inches away for the physical robot. Task should be completed in less than 3 minutes. Your algorithm should perform correctly for either left or right turns. Do not combine left with right turns. During evaluation, the TA will start the robot at different initial positions and orientations.

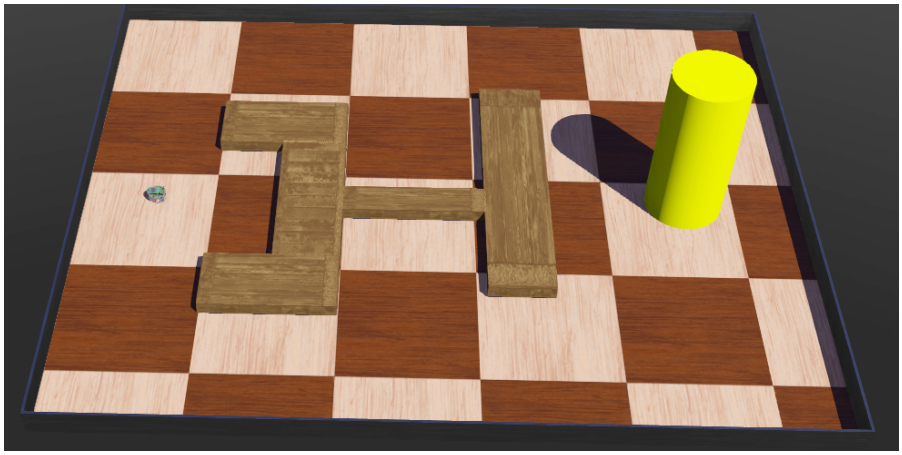

Figure 2. Robot simulator world configuration for bug algorithm.

D. Task Evaluation

Task evaluation is based on: (1) program code, (2) report, including a link to a video showing each different task, and (3) task presentation of robot navigation with the TA. Note that all functions and tasks to be implemented are required in future labs.

D.1 Task Presentation (90 points)

The following section shows the rubric for the different tasks:

- Task 1 (45 points)
  - Robot reaches the cylinder from any starting position and orientation (20 points)
  - Robot finds the cylinder when the cylinder is not seen in the camera (20 points)
  - Robot stops 10 inches or less from the cylinder (5 points)
  - Robot hits the cylinder (-5 points)
- Task 2 (45 points)

- Robot completes correct bug zero algorithm stopping within 10 inches of the cylinder (20 points)
- Robot moves correctly around walls (10 points)
- Robot switches correctly from motion to goal to wall following (5 points)
- Robot switches correctly from wall following to motion to goal (5 points)
- Robot performs consistent left or right turns (5 points)
- Robot hits the cylinder (-5 points)
- Robot travels a hardcoded path defined in advanced (-10 points)

D.2 Task Report (10 Points)

The report should include the following (points will be taken off if anything is missing):

1. Mathematical computations for all kinematics. Show how you calculated the speeds of the left and right servos given the input parameters for each task. Also, show how you decide whether the movement is possible or not. (4 point)
2. Video uploaded to Canvas showing the robot executing the different tasks. You should include in the video a description, written or voice, of each task. You can have a single or multiple videos. Note that videos will help assist task evaluations. (3 point)
3. Conclusions where you analyze any issues you encountered when running the tasks and how these could be improved. Conclusions need to show an insight of what the group has learnt (if anything) during the project. Phrases such as "everything worked as expected" or "I enjoyed the project" will not count as conclusions. (3 point)

D.3 Task Presentation

Task presentation needs to be scheduled with the TA. Close to the project due date, a timetable will be made available online to select a schedule on a first come first serve basis. Students must be present at their scheduled presentation time. On the presentation day, questions related to the project will be asked, and the robot's task performance will be evaluated. If it is seen that any presenter has not understood a significant portion of the completed work, points will be deducted up to total lab points. Students that do not schedule and attend presentations will get an automatic "0" for the lab.

NOTE for physical robot presentation: During presentations, robot calibration time is limited to 1 min. It is important that all members of the team attend and understand the work submitted.

E. Lab Submission

Each student needs to submit the programs and report through Canvas as a single "zip" file.

- The "zip" file should be named "yourname_studentidnumber_labnumber.zip"
- Videos should be uploaded to the media folder in Canvas. Name your files "yourname_studentidnumber_labnumber_tasknumber".
- The programs should start from a main folder and contain subfolders for each task.
- The report should be in PDF and should be stored in the same "zip" file as the programs.