Exercise 2: Intro to Open CV*

EENG450AB: Systems Exploration, Engineering, and Design Laboratory

Vibhuti Dave and Tyrone Vincent

Department of Electrical Engineering
Colorado School of Mines

Spring 2018

1 Introduction

The objective of this module is to introduce you to OpenCV. This will help give the robot the gift of vision enabling it to detect beacons and react accordingly. The robot will need to

- Take pictures in real time
- Process these pictures to eliminate all background information and noise. The only part of the picture that is of interest is the beacon and order of LEDs on the beacon
- Figure out the specific beacons detected
- Locate itself with respect to the location of the beacons. This is where the robot will need to know its position (coordinates) in a grid and move to a position relative to the position of the beacons. The position of the beacons will be known. Once you know where and which beacons your robot can see, it should be able to know its own position and then move to a location specified.

This tutorial is meant to get you to the point where you are able to detect different colors and shapes in an image, and perform some processing that removes background noise and cleans up the image.

2 Setting up the Camera

Your Raspberry Pi provided comes pre-installed with OpenCV. To get started:

- Shut down the Pi and connect the camera module to the Pi.
- Open up a terminal and navigate to your "projects" folder.
- Make a subfolder called opencv. Save all your programs pertaining to image processing in this folder for organizational purposes.
- OpenCV has been installed in its own virtual environment. To get access to all the libraries and methods available with OpenCV, you will need work in the OpenCV environment. Every time a new terminal window is opened, use the following at command prompt
  - `source ~/.profile`
  - `workon cv`

- Examine the command line to ensure that you are working in the cv environment. If you see the text "(cv)"preceding your prompt, then you **are** in the cv virtual environment.
- At command prompt, use `pip install "picamera[array]"`

You can use the Python IDLE 2 application to type up your py code and run it from there[1]

Note: If you ever buy your own Raspberry Pi and are interested in installing OpenCV from scratch on your Pi, follow these instructions.

3 Pi Camera and OpenCV Turorial

Here is a beginner's tutorial to capturing an image, recording video, and applying some image effects to the pictures you take. Here is a set of tutorials to use OpenCV with Python. **Right now** go through **at least** the tutorials on "GUI features for Open CV" (skipping video), and "Core Operations". You should also bookmark this page and go back to it as necessary as you need more features from Open CV. Note that an image is stored as an array of pixels. Numpy is an optimized library for fast array calculation. Refer to this Python-Numpy tutorial. There is an example code called take_picture.py saved in the home directory. You can use this as a starting point.

4 Camera and OpenCV Exercises

For all of the exercises mentioned below, it is necessary to work in the opencv environment. Some of the libraries that need to be imported include `numpy, cv2`, and `picamera`. There will be no access to the methods and functions of opencv unless you are in the OpenCV environment.

1. Write a Python script that does the following
   a. captures an image
   b. asks the user for a filename to store the image
   c. stores the captured image using the filename provided by the user.
   d. displays the image on the screen
   e. asks the user to select a pixel using the mouse and displays the 3 color values for that pixel

   Your program and image will be in the same directory. Refer to PiCamera Docs and getting started with images to learn about functions you can use to complete the exercise. For efficiency, I recommend capturing and storing the image as an OpenCV object. Look in to using the classes in `picamera.array`, specifically `PiRGBArray`.
2. Write a Python script that reads a previously stored image (use an image captured from part 1), resizes the image to half its size without changing the aspect ratio (Hint: Use `cv2.resize`). Refer to geometric transformations of images to get details on scaling an image. Example: a 1280 X 1118 should be resized to 640 X 559.
3. Download `Colors.pptx` from Canvas and display it on a monitor. Write a Python script that does the following
   a. captures and image (The slide will be the object you will take the picture of)
   b. stores the image as colors.jpg
   c. makes a copy of the original image
   d. makes the original image smaller

[1] If the CV environment is not found, try copying the cv2.so file from /usr/local/lib/python2.7/site-packages to your project directory.

e. detects the color yellow in this image. **Our recommendation is to change color spaces from BGR to HSV** using `cv2.cvtColor`. The only thing visible should be the color yellow in the modified image with everything else black. You may want to read about the HSV color space using resources from the internet. Also, be use to understand how Open CV saves HSV values for 8-bit images (H is divided by 2 to fit between 0 and 255).

f. Display the original image and the image after the color yellow has been isolated side by side.

This blog post practically gives you the solution. Note, the blog post mentioned above uses the BGR color space. In the HSV, color space, the lower and upper bounds will need to change.

4. Take three pictures (using the Pi Camera) of the colors slide in different lighting conditions, at different distances and at different heights. Change the images from BGR to the HSV color space. This time isolate yellow, blue, green, and red in all of these images. You may want to use your color picking program that you created above to determine appropriate limits. Save the images with the isolated colors for use in the next exercise.

   For real-time images, it is a good idea to change the auto white balance of the camera. The white balance is set to auto by default. You may wish to use settings to ensure consistent images.

5. Choose one of the saved images from exercise 5 with the different colors isolated. Clean up the image by performing morphological transformations. Refer to morphological transformations to get details on methods available for image processing. Experiment with all the methods to see the various effects. It is possible to run more than one transformation on the same image. Example the image can be opened first and then closed. Run some experiments to see what works best with the image you chose. Save the best result.

6. Convert the cleaned-up image to a grayscale image using cv2.cvtColor. Next, filter out any noise in the image using cv2.GaussianBlur and save.

7. Use cv2.SimpleBlobDetector_Params() and cv2.SimpleBlobDetector_create(params) to detect the 4 blobs (shapes) in the image from part 6 and draw contours around the 3 blobs. These functions allows you to filter blobs based on the following criteria:

   a. By color

   b. By size

   c. By shape (circularity, convexity and inertia ratio)

   The official opencv page provides the details on the function and the parameters for C++. Two great examples on how to use the functions in python and draw green circles around the detected blobs can be found here and here.

5 Deliverables

Create a document that answers these questions

1. How is an OpenCV image stored?
2. What does `image.shape` return? Here image is the variable name that stores the image you are referring to.
3. What do the parameters, `fx` and `fy` refer to in `cv2.resize`?
4. What would happened if, `cols/2`, and `rows/2` in the following function were changed to `cols/4` and `rows/4`?

   ```
   M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
   ```

5. What function must be used after `cv2.getRotationsMatrix2D` to actually perform the rotation? What parameters do you pass to this function?
6. What are the different morphological transformations that can be performed on an image?

7. What effect does the morphological transformation, “opening” have on an image?
8. What effect does the morphological transformation, “closing” have on an image?
9. What does HSV stand for?
10. What do the upper and lower bounds in exercise 4 represent?
11. A kernel of ones was created before performing morphological transformations. What is the effect of changing the size of the kernel?
12. How would you use the `simpleblobdetector` functions if you only wanted to detect the circular blob while ignoring the rest?

Save a copy of your well documented code and results (i.e., image files, etc) so that you can refer to these later. On your group page on Canvas, under Files, create a folder called “Project 2 Computer Vision”. Upload your code, results and document with answers to the questions above to that folder.

6 Demonstration

Be prepared to demonstrate your results, explain your code, and modify your code to achieve specific goals.