2.12: Introduction to Robotics
Lab 8:
Computer Vision*

Spring 2023

Assigned on : 4-6-2023 Due by 4-12-2023

Instructions:

1. **Feel free to work alongside fellow classmates, but turn in your own work.**
2. **Submit a PDF of the collection of screenshots and answers to the questions that are asked for in this lab to Stellar.**

1 Introduction

In this lab, you will experiment with several computer vision techniques. Specifically, you will:

1. Explore gamma adjustment.
2. Explore Otsu's method of thresholding.
3. Explore Canny edge detection.
4. Explore Hough transforms to find circles and lines on an image.
5. Look into the difference between using HSV and RGB to detect objects.
6. Explore morphological operations of dilation, erosion and combinations of the two.
7. Explore Lucas-Kanade Optical Flow.

We will use OpenCV (Open Computer Vision) library to process images captured by your webcam.

*

1. Version 1 - 2016: Peter Yu, Ryan Fish and Kamal Youcef-Toumi
2. Version 2 - 2017: Luke Roberto, Yingnan Cui, Steven Yeung and Kamal Youcef-Toumi
3. Version 3 - 2019: Jerry Ng, Jacob Guggenheim
4. Version 4 - 2020: Jerry Ng, Rachel Hoffman, Steven Yeung, and Kamal Youcef-Toumi
5. Version 4 - 2020: Phillip Daniel

2 Setting up

Open a new terminal in your machine, and enter the following commands to download the things that you need for the lab. You will be double checking that you have the proper version of many of the software tools installed, and then running the installation codes just to be certain that your software is up to date. We noticed that at times Ubuntu will report only a subset of its installed software versions if multiple versions are installed. This is particularly true of Python 3.

2.1 Python

First confirm if you have the proper version of Python installed. Run this command to check if you the latest Python3 version is installed.

```
python3 -V
```

2.2 Matplotlib

```
cd ../../.. # Move out off the lab1 directory
pip install matplotlib #Install matplotlib
git clone https://github.com/mit212/lab8_2023.git #Clone the github
cd lab8_2023  # go into the lab8 directory
```

Separately, we ask that you choose an object, preferably one of a solid color, to use for the entire lab. Throughout the lab, we will attempt to separate that object in the images taken by your camera from everything else.

Make sure your camera is connected to your computer before using any of the scripts. OpenCV will automatically open the primary camera connected to your computer.

3 Gamma adjustment

In computer vision, one of the first things introduced is gamma adjustment or correction. To experiment with this concept, run the following command:

```
python3 gammaAdj.py
```

When you run this command, three windows will appear. One will show your unprocessed raw camera footage, another will show a processed (gamma adjusted image), and the last will have a slider. Play around with the slider and see what you observe (make it completely black or white).

4 Otsu's method of Thresholding

It is important to note that one of the prime goals of computer vision is to separate an image into *what you care about* and *what you don't care about*. Otsu's method of thresholding is a prime example of a simple method that algorithmically separates pixels into two separate classes.

To use it, run the following command:

```
python3 otsu.py
```

When you run this command, a single image will be taken and then processed using Otsu's method of thresholding. A histogram will appear on your screen, as well as the unprocessed and processed image.

5 Canny Edge Detection

Though Otsu's method is nice as it comes directly as an output of a relatively fast algorithm, it is apparent that more complex techniques need to be developed so that we can tackle this problem of separating *what you care about* and *what you don't care about*.

To that end, Canny edge detection is an attempt at answering this problem when all you care about is **edges**. Hypothetically, as your object has edges, you would be able to separate it from its background if it is an edgeless background.

To experiment with the concept further, run the following command:

```
python3 canny.py
```

You'll notice that if you increase the lower threshold above the upper threshold, the algorithm should automatically switch so that the lower threshold becomes the upper threshold for the algorithm.

6 Hough Transforms

Similarly to how Canny Edge detection would work great if your object is the only object with edges, Hough transforms is great if your object is either circular or a line.

6.1 Lines

To detect lines, run the following command:

```
python3 houghLines.py
```

Unlike the circles algorithm, the lines algorithm is much less time intensive when it comes to false negatives so there is no image shrinking done for this one.

6.2 Circles

To detect circles, run the following command:

```
python3 houghCircles.py
```

Due to how many false circles are detected by the algorithm, the frame is shrunk with respect to the original captured image. This is accounted for in the accumulator ratio slider bar. If you wish to play more with this, you can change the ratio of the shrunk image with respect to the ratio.

More information on each command can be found here:

1. Circles: `https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html`
2. Lines: `https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html`

6.3 Sudoku Grid

Before running this on camera footage, you will be processing the sudoku grid image from lab 1. Namely, you'll be editing the file `process.py`. The goal for this script is to isolate the grid lines and only the grid lines of the sudoku grid. Upon completion, take a screenshot of the line detected grid output of the Hough Transform.

7 HSV vs RGB

If you were paying very close attention, you may have noticed that **ALL** the algorithms before this section are conducted after converting your raw image into grayscale. With that in mind, we will now experiment with colors. For this section, in addition to the object you've been using, we ask that you find a few different colored objects to experiment with the robustness of the two methods for separation presented here.

To begin the script, run the following command:

```
python3 colorThresh.py
```

After capturing the objects of your choosing, you'll want to note down the HSV values for the next part of the lab.

8 Morphological Operations

After using the color thresholding, you may have noticed that there are some features that remain in the image aside from your object that you want to detect. With that in mind, morphological operations are an excellent way of getting rid of extra features you don't want and enlarging objects you do want.

To experiment and find which operations work best, run the following command:

```
python3 morphOps.py
```

Something else interesting about morphological operations is the kernel operator that is used. If you open the script, you'll find a section of code that allows you to write in your own kernel (a vertical line is given as an example in the comments). Rerun the script with a kernel of your own design and take a screenshot for your submission of it in action.

9 Optical Flow

Optical flow has several uses. The one we'll present here is tracking an "object" as it moves through space. Hypothetically, if you were able to place trackers based on the output of your morphological operations presented before, you'd be able to use optical flow to track your object in space. In this case, we will be placing trackers on detected corners.

To run the optical flow script, run the command:

```
python3 opticalFlow.py
```

To tune the features tracked by the script, you can edit the variable `feature_params` and decrease the `qualityLevel` and increase the `maxCorners`. Tune these features so that when running this script, a corner is detected on your object.

10 Conclusion

Upon completion of this lab, turn in the screenshots you took throughout the process and the answers to all of the 9 questions asked. There should be a total of four different screen shots:

1. Otsu output image
2. Hough line transform of the sudoku grid

3. Isolated object after morphological operations
4. Custom kernel