Lab 6 – Basic Computer Vision

Introduction

Vision is a very powerful ability that allows us to interact and navigate our world. Implemented on a robot, it can greatly expand the robot's capability. But vision can also be very difficult to program.

Overview

In this Lab you will be experimenting with different image processing techniques. You will first implement the processes by hand and then through the OpenCV methods.

Outcomes:

- Familiarity with image processing techniques
- Familiarity with OpenCV
- Tracking an object of interest in an image

Procedure

0. Setup
    a. Install OpenCV
        i. You can install it in PyCharms by creating a new project called Lab6. Then go to File>Settings>Project:Lab6. Under the python interpreter click the '+' to add a module. Search for opencv-python
        ii. You can also install OpenCV on your computer system wide through pip
            1. pip install opencv-python
    b. Download and run the example script and images
    c. Run the example script for both an image and your laptop's camera.
1. Image Segmentation
    a. Load testImage1.png with the example script
    b. Use the mouse callback in the example script, get a color reading from the dark blue shape
    c. Create a lower and upper bound that encompasses the color reading
    d. Do an image segmentation where all the dark blue pixels in the image are turned white and everything else is black. Do not use OpenCV methods
        i. Note, that the input image has the shape (height,width,3). The segmented image should have the shape (height,width,1)
    e. Find the centroid of all the white pixels and draw a small circle in the image at that location
    f. Repeat the prior steps but this time use OpenCV's inRange() method
        i. https://docs.opencv.org/3.4.15/da/d97/tutorial_threshold_inRange.html
        ii. Note, that the dimensions of the image from inRange should be (height,width,1), but if you do np.shape() you will only get (height,width). In

numpy, if the last dimension is 1 it doesn't properly show up. You will have to reshape the matrix returned by inRange() to be (height,width,1) to get the 3rd dimension back. **This reshaping goes for any matrix returned by an OpenCV method.**

g. For all your results, show and compare your hand coded solution to the OpenCV solution
h. For each subsequent step, use the resulting OpenCV solution not your handed coded solution

2. Dilation and Erosion
    a. Apply a dilation to the OpenCV segmented image from Task 1. Do not use the OpenCV dilate method.
    b. Apply an erosion to the OpenCV segmented image from Task 1. Do not use the OpenCV erosion method.
    c. Apply an opening (dilation followed by erosion) to the segmented image from Task 1. Do not use the corresponding OpenCV methods.
    d. Find the centroid of all the white pixels and draw a small circle in the image at that location
    e. Repeat the prior three steps with the dilate() and erode() OpenCV methods. Compare your hand coded solution to that of OpenCV.
        i. https://docs.opencv.org/3.4.15/d9/d61/tutorial_py_morphological_ops.html
3. Edge detection (Laplacian and Sobel)
    a. Apply the 4 neighbor Laplacian operator to extract all the edges
    b. Apply a horizontal Sobel operator to the Laplacian image to extract the horizontal edges
    c. Apply a vertical Sobel operator to the Laplacian image to extract the vertical edges
    d. Find the centroid to the Laplacian image and draw a small circle in the image at that location
    e. Repeat the prior steps with OpenCV methods. Make sure to look at the document for the parameters that correspond to the direction and kernel sizes shown during lecture
        i. Laplacian and Sobel - https://docs.opencv.org/3.4.15/d5/d0f/tutorial_py_gradients.html

0	-1	0
-1	4	-1
0	-1	0

Figure 1 Laplacian operator (4 neighbors)

4. Track in camera
    a. Using the example code as an example, grab the frame from a live camera
    b. Pick a colorful object like a balloon or colored sheet of paper to hold infront of the camera
    c. Get color readings for your object and create a lower and upper bound
    d. Apply the segmentation and opening operations to the frames from the live camera feed
        i. Draw the centroid on it
    e. Move the object and see if the centroid properly moves with it
5. Connect Components

a. Repeat task 4 but this time use OpenCVs connectedComponentsWithStats to track the largest blob in the image. In theory, the largest blob should be your object. You do not have to code Connected Components by hand.
    i. https://www.pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/

Questions

1. The OpenCV methods are going to be much faster than your methods. What are some ways to increase the speed of your operations?
2. For task 2, you applied the Laplacian to obtain all the edges. You then applied a horizontal and vertical Sobel to get the horizontal and vertical edges? Is there any advantage to being able to differentiate the horizontal and vertical edges out of all the edges?
3. (Double points) Explain how Connected Components work.

Write up Reminders

1. Methods
    a. Explain the how the different processing methods you implemented work
    b. Explain you hand coded solutions
    c. Brief overview of working with OpenCV and loading/viewing images or live camera feed
2. Results
    a. Screenshots of your image manipulations for the various steps. Be sure to include the original images with the transformations
3. Supplementary Material and Submission
    a. Submission – One zipped folder named after the team members usernames. In the folder should be the report and a folder for your code
    b. Supplementary Material – Should contain the following sub-sections
        i. Section 1
            1. List of python files you submitted and a 1-2 line description of each
        ii. Section 2
            1. Screenshots/pdfs of your code
    c.

Rubric – Lab 4

General formatting (10)

- Title, date, name, section headings
- Miscellaneous style and formatting

Abstract (15)

- Gives a brief intro or overview of the Lab and/or topics covered.
- Gives a brief summary of the Results and main points from the Discussion.
- Well written and concise without typos or grammar issues.

Methods (20)

- Possess all relevant equations and algorithm explanations
- Tasks and procedures are described.
- Looks nice and is easy to follow.

Results/Supplementary (40)

- Figures and text for all experiments.
- Data and analysis for all experiments. Includes any required statistical summaries or tests.
- Figures are properly formatted (labels, legends, captions, etc.).
- Required videos and code.

Discussion (15)

- All questions are answered.
- Well written and concise without typos or grammar issues.