EXPERIMENT #3
BASICS OF IMAGE PROCESSING WITH JETSON NANO

Instructor: Dr. Jafar Saniie
ECE 498-07

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature: Maggie Barclay & Theo Guidroz

Introduction

A. Purpose

In this lab, students will run the same example as in lab one using python and the jetson nano this time.

The purpose of this lab is to introduce the following items:

- Python compatible image processing library - OpenCV
- Jetson Nano specific image processing toolkit - CUDA
- Realization of image processing techniques in Experiment #1 on Jetson Nano

B. Background

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed (a family of permissive free software licenses)product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. In terms of Jetson Nano, it supports OpenCV with GPU acceleration. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

The NVIDIA® CUDA® Toolkit is used to develop, optimize and deploy GPU-accelerated apps on products with a NVIDIA GPU, including Jetson Nano. It provides a development environment for creating high performance GPU-accelerated applications. With the CUDA Toolkit, you can develop, optimize and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms and HPC supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler and a runtime library to deploy your application.

GPU-accelerated CUDA libraries enable drop-in acceleration across multiple domains such as linear algebra, image and video processing, deep learning and graph analytics. For developing custom algorithms, you can use available integrations with commonly used

languages and numerical packages as well as well-published development APIs. Your CUDA applications can be deployed across all NVIDIA GPU families available on premise and on GPU instances in the cloud. Using built-in capabilities for distributing computations across multi-GPU configurations, scientists and researchers can develop applications that scale from single GPU workstations to cloud installations with thousands of GPUs.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- 1 x Jetson Nano single-board computer
- Internet connectivity
- I/O devices (USB keyboard/mouse, etc.)
- External Monitor with HDMI/DP port

B. Procedure

Part A

1. Run the image segmentation source code with only the thresholding technique applied
2. Check results from the same script, this time using your own image.
3. Create and run the edge detection script.

Part B

1. Run the shape extraction program with the test image
2. Add noise to the test image
3. Modify the shape extraction program to run with the image with added noise. Check the results.
4. Redo step 2 and 3, but with a variance of 0.01

Part C

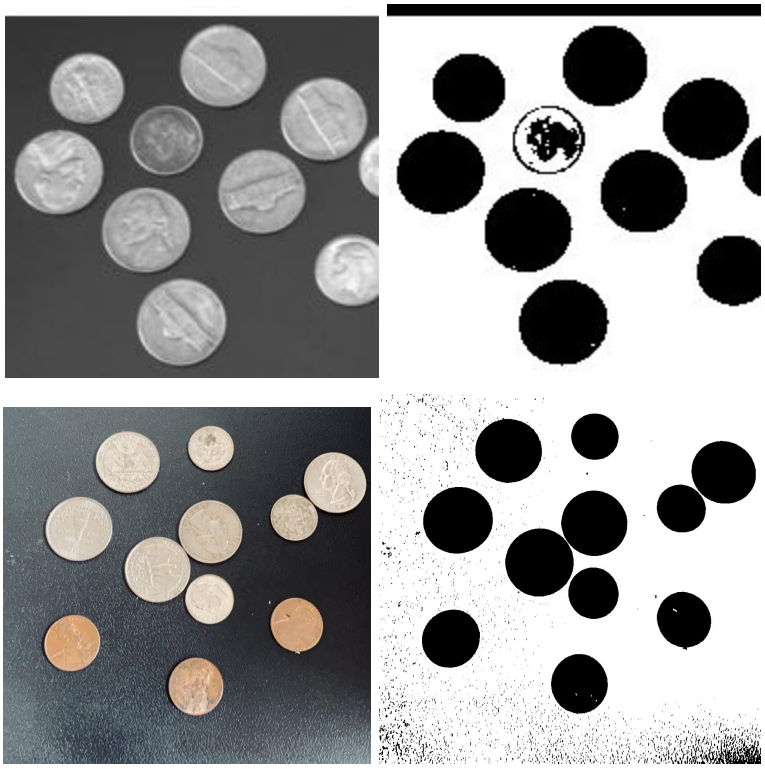
1. Try opening, closing, erosion, and dilation with Python (OpenCV)
2. Use your own image to perform histogram expansion and compare the result with the example effect image in the background section.

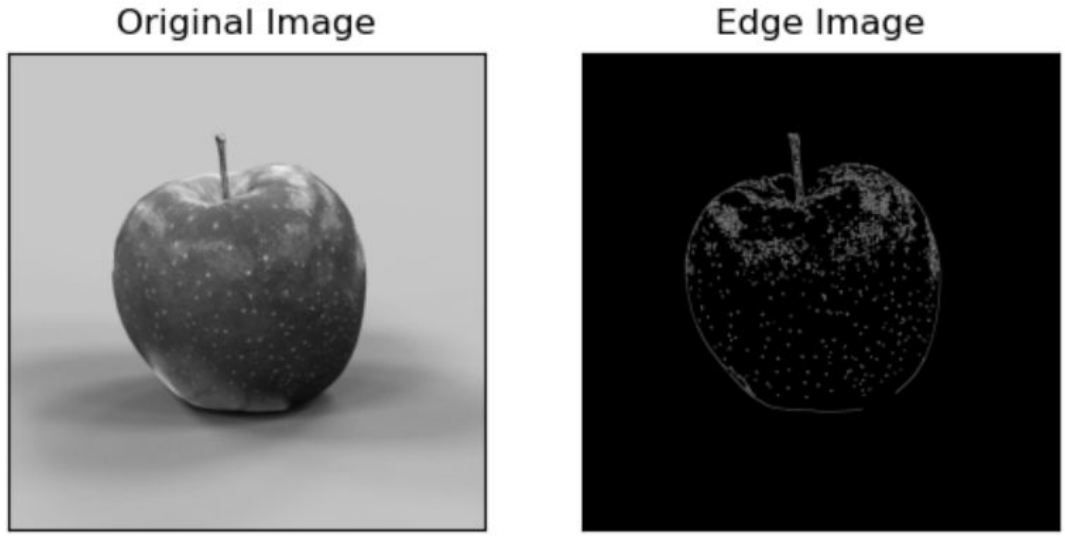
Part D

1. Create and run the video processing program with the test video
2. Create a program to use the shape extraction program in part B to process these frames iteratively
3. Use generate video from frame function to create a video processing program to generate a video with labeled shapes

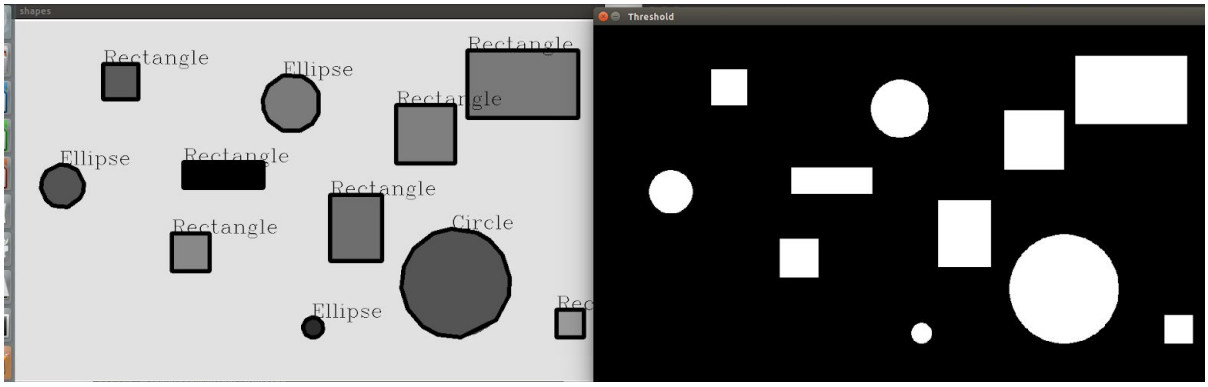
III. Results and Analysis

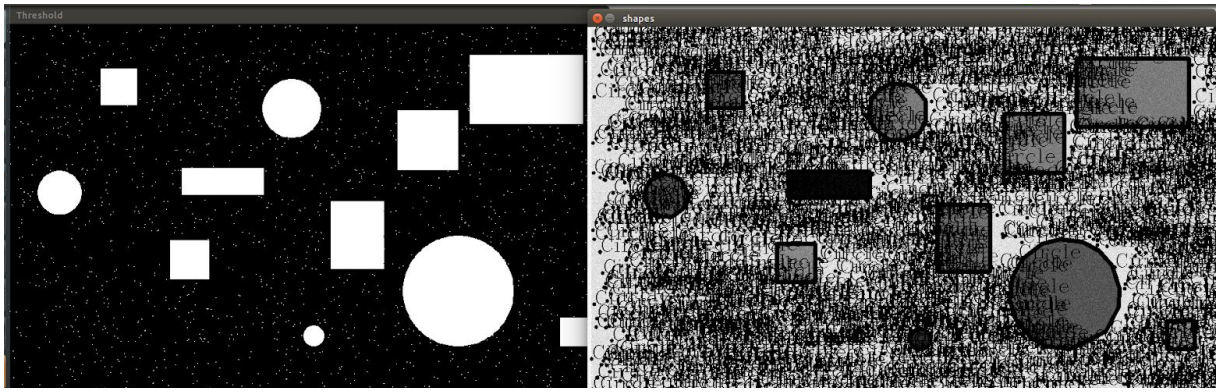
PART A - Image segmentation





PART B


shapes
Rectangle
Ellipse
Rectangle
Rectangle
Ellipse
Rectangle
Rectangle
Circle
Ellipse
Threshold


Threshold
shapes

Part C


Ope
Dilatio
Erosion
Closing


Equalize

Part D

Students referred to the code from the following github to extract the frames:
https://github.com/TheCatLover/Video-Frame-Processing

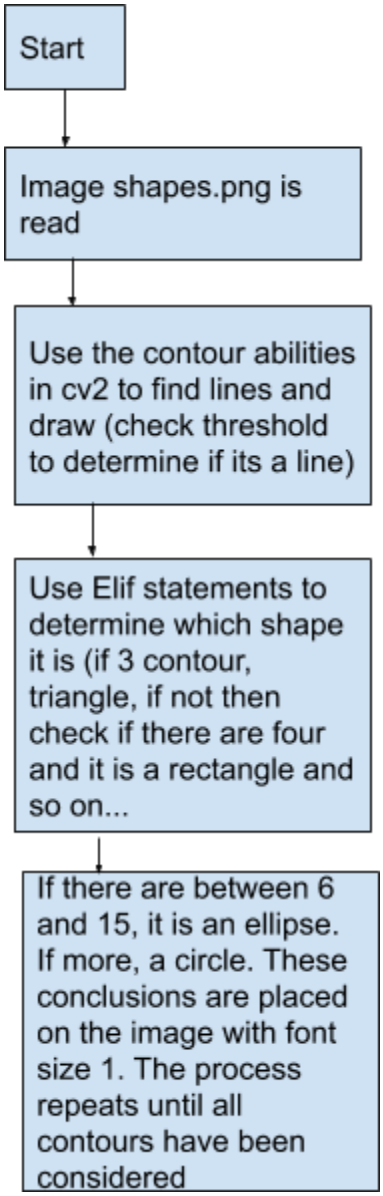
Result of the shape extraction:

Questions

1. Comment on the advantages and disadvantages of programming with MATLAB and Python, specifically on the image segmentation script/program. (10 pts)

In the image segmentation program for both python and MATLAB, both programs seemed to be successful in identifying the coins. There are clear circles where the coins are, and they are clearly discerned from the background. However, the most notable disadvantage of the MATLAB code is that with the personal image we used, it seems to be sensitive to noise compared to the python script.

2. Draw a flow chart and explain how the shape extraction program in Appendix A works. (10 pts)



3. Why are the numbers different between the MATLAB and Python noise generation script/program? (10 pts)

The languages implement noise in different ways. The MATLAB script uses 'imnoise' function which takes the mean and variance. Python does this differently, as the variance is multiplied by another number to get sigma and this value is fed into the function. (I might be misinterpreting this question, though I think the answer is just that the functions creating the noise have two distinct sets of parameters.

4. Why is GPU acceleration useful/beneficial to image/video processing in general? (10 pts)

GPU acceleration allows for faster training of data (from our personal experience with the project).

5. In the video processing part, explain why we need to generate separate frames to perform video processing. Assume the video is encoded. (10 pts)

Each frame from a video will be separately processed. The way in which we have seen edge detection, image classification, and similar techniques work is by feeding an image in. This will be done in the same exact way with a video, therefore we need to take frames and run it through these same programs.

IV. Conclusions

In this lab, students built a deeper understanding of the Jetson Nano and how to set it up. The sample programs were tested and the results were as expected. Overall, this lab was successful.

References

[1] Lab 3 manual
[2] NVIDIA CUDA - Develop, Optimize and Deploy GPU-accelerated Apps
https://developer.nvidia.com/cuda-toolkit
[3] Image Segmentation with Watershed Algorithm
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html