Image Processing HW1

tags: `2019imageprocess` `hw`

url https://hackmd.io/qKRGmVSWS5KX4w8zSVwT8w (https://hackmd.io/qKRGmVSWS5KX4w8zSVwT8w)

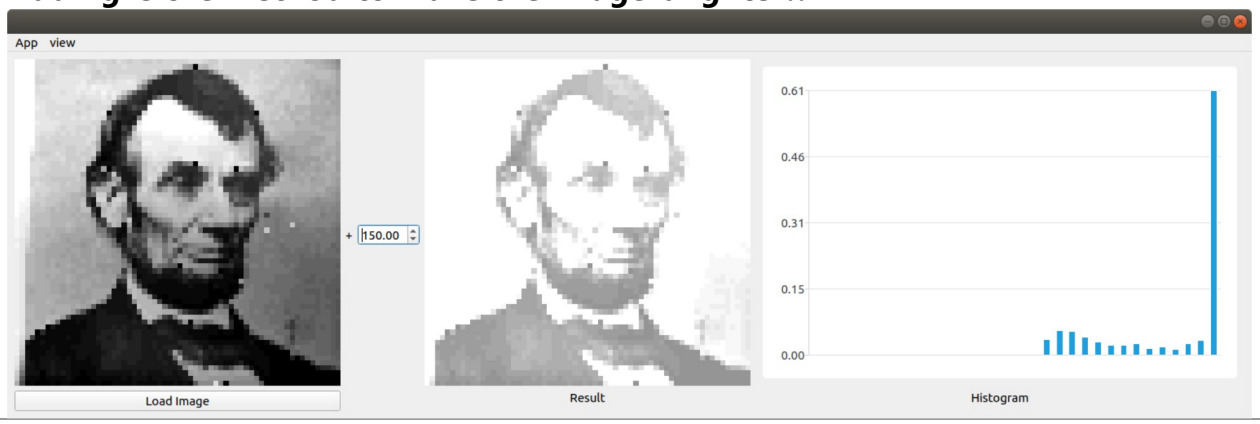
github https://github.com/linnil1/2019ImageProcessing/tree/master/hw1 (https://github.com/linnil1/2019ImageProcessing/tree/master/hw1)

Homeworks

- Process the image array to obtain the histogram of the image.
- hw1-1 imageBasic

> Read the special .64 image file into a 64x64 image with 32 gray levels and store the data in a 2-dimensional array.

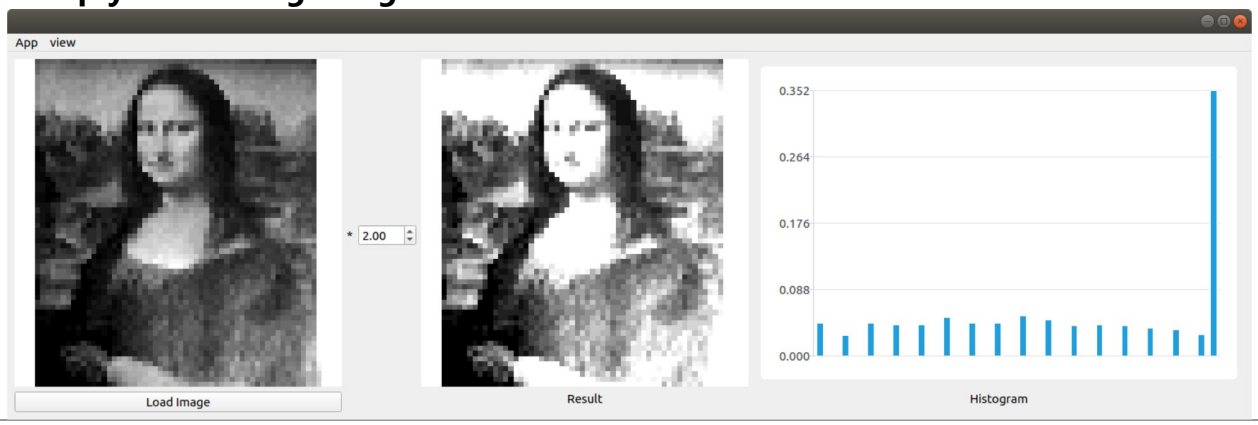
- hw1-2 Arithmetic Operations of an Image Array
- hw1-2-1 imageAdd

> Add or subtract a constant value to each pixel in the image.
>
> **Adding is the method to make the image brighter!!**
>
> 
>

- hw1-2-2 imageMult

Multiply a constant to each pixel in the image.

Muliply make image brighter but also add contrast



- hw1-2-3 imageAvg

Create a new image which is the average image of two input images.

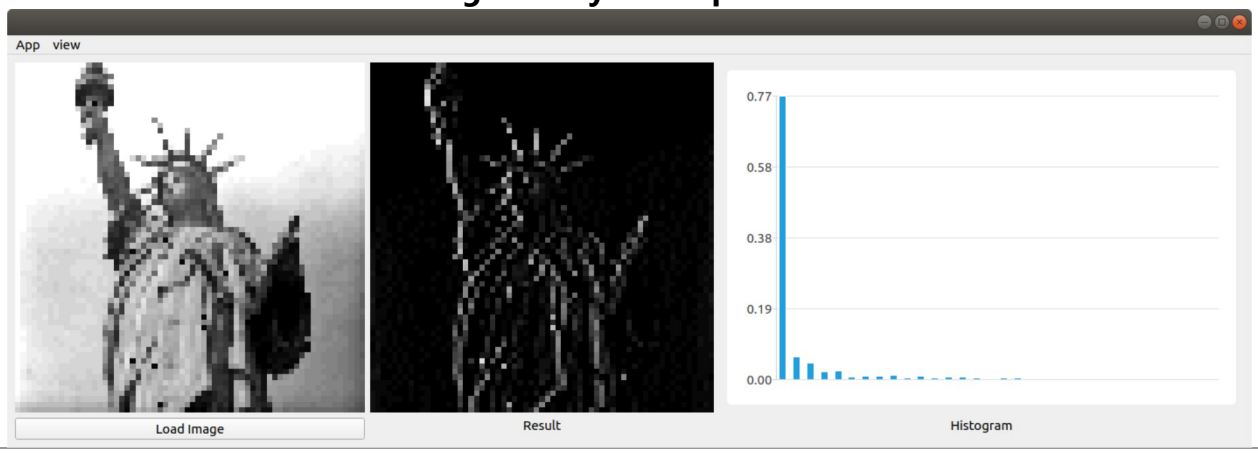
Maybe it's the way to merge two image

- hw1-2-4 imageSpecial

Create a new image g(x,y) in which the value of each pixel is determined by calculating the pixel values of the input image f(x,y) using the following equation:

`g(x, y) = f(x, y) - f(x - 1, y)`

This equation is a little bit similar to derivative in x axis, the result show that we can find out where the edge are by this equation



My solutions

I write python code `hw1_np.py` with tool `numpy`. However, using it is a little bit easy, so I write another c++ extentsion for python called `MyImg.h`, `py_myimg.cpp` and `hw1.py`.

To show the result, `QT` or `matplotlib` is used, you can run `qt.py`, `hw1_np.py` respectively.

Requirement

- Ubuntu18.04 (Debian like is OK)
- Python3.6.8
- Qt5.13.1

You can install with below commands.

```
./setup.sh
```

```
sudo pip3 install -U numpy matplotlib PyQtChart
sudo apt update
sudo apt install -y python3-pyqt5
```

Using Docker(Alternative)

Setup your docker step by step with this link (https://docs.docker.com/install/linux/docker-ce/ubuntu/).

Build the dockerfile

```
cd hw1
docker build -t linnil1/2019imageprocess .
```

Run it

```
docker run -ti --rm \
    --user $(id -u) \
    -v $PWD:/app \
    -v /tmp/.X11-unix:/tmp/.X11-unix \
    -e DISPLAY=$DISPLAY \
    linnil1/2019imageprocess python3 qt.py
```

Usage

The working dir is at `hw1` .

Without QT, with numpy

This program read from commandline and plot by matplotlib.

e.g. To solve hw1-2-1, you can enter `python3 hw1_np.py data/LINCOLN.64 --add 200` or `python3 hw1.py data/LINCOLN.64 --add 200`

Use `python3 hw1_np.py --help` to get more information.

Note that even the original image is 32-level gray scale, I remap it to 256 scale, so the maximum of addition is 256.

Without QT and numpy

Still, plot by matplotlib.

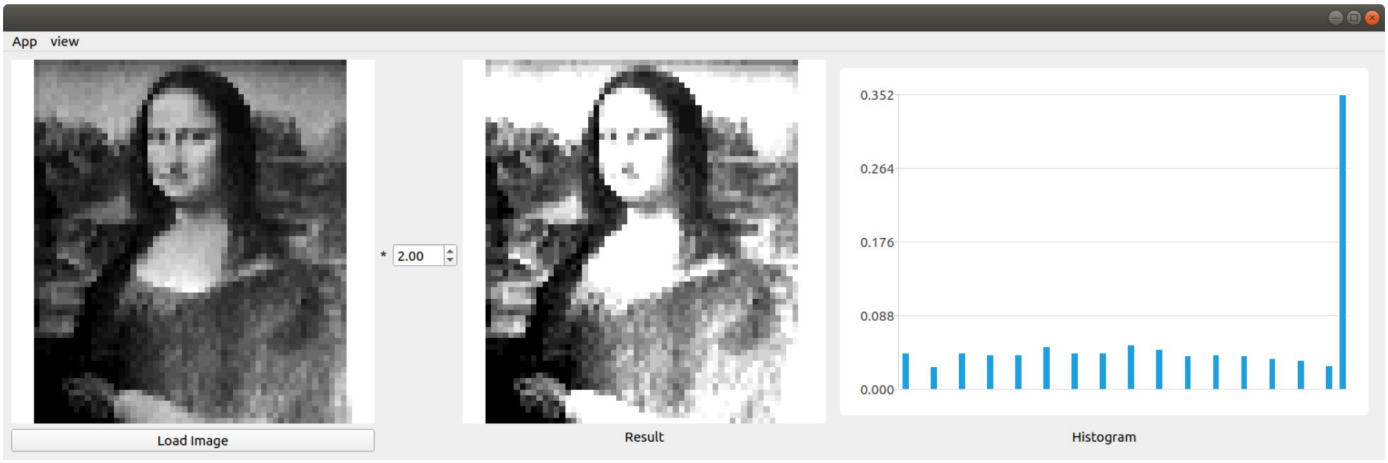
Build it `python3 setup install`

and run `python3 hw1.py data/LINCOLN.64`

With QT(Default)

`python3 qt.py` or `./run.sh`

The promgram should work like this.



Algorithm of histogram

I normalize any image to 0~1 and rescale back to 32-levels, then round it to interger. The interger is the index of array and add unit if appear once, two for two times, and so on.

The python code

```
arr = np.array([])
arr.resize(32)
map32 = np.uint8(img * 31).flatten()
for i in map32:
    arr[i] += 1
return arr / arr.sum()
```

心得

這次作業有兩個難關要走，一個是 c++ extention for python，要讀 python 的 c-api 還要把他 build 成 python module. My Note (https://hackmd.io/PUAGCBSyQTSU04_pd_qOrQ)

第二個是 UI， UI 用 QT 寫的難度很高(因為 QT 多半是 c++ 的 document)，不像原本 matplotlib(python) 的這麼好用。

第三點是 還要把東西包成 exe 檔，實在不懂，全世界的服務幾乎都跑在 linux 上，造成麻煩而且也有安全性問題GG

這次的 algorithm 很少，就是兩個迴圈硬幹就對了。在此就不贅述。剩下的技術(語法技巧)就寫在註解了。