Image Processing HW6

tags: `2019imageprocess`

github url: https://github.com/linnil1/2019ImageProcessing/tree/master/hw6 (https://github.com/linnil1/2019ImageProcessing/tree/master/hw6)

hackmd url: https://hackmd.io/8cPlymEeSUuQW8QvAdid-w (https://hackmd.io/8cPlymEeSUuQW8QvAdid-w)

Wavelet Transform

First, try 1D wavelet transform as the example in textbook.

The code is easy to implemented.

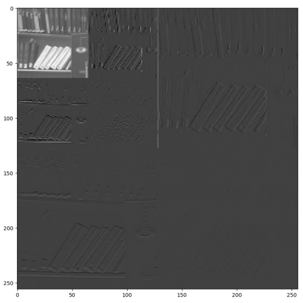
```
result = []
for i in range(J):
    result.append(np.convolve(data, harr_wa
    data = np.convolve(data, harr_scale_r)[
result.append(data)
result = np.concatenate(list(reversed(resul
```

Then, we get `[ 1. 4. -2.12132034 -2.12132034]` , which is agree with answer.

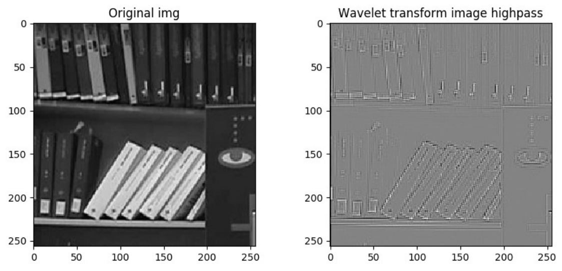
Then, try 2d wavelet transform by recursion.

```
def wavelet2D(data, depth):
    if not depth:
        return data
    wavel = convolve(data, harr_wavel_r)[:,
    scale = convolve(data, harr_scale_r)[:,

    wavel_wavel = convolve(wavel, harr_wave
    scale_scale = convolve(scale, harr_scal
    wavel_h =     convolve(wavel, harr_scal
    wavel_v =     convolve(scale, harr_wave

    scale_scale = wavelet2D(scale_scale, de
    return np.vstack([np.hstack([scale_scal
                      np.hstack([wavel_v, w
```

Work as charm



Remove the low frequency image and inverse back.



See detail in `hw6/test_wavelet.py`

Part 1: (30%) Geometric Transformation

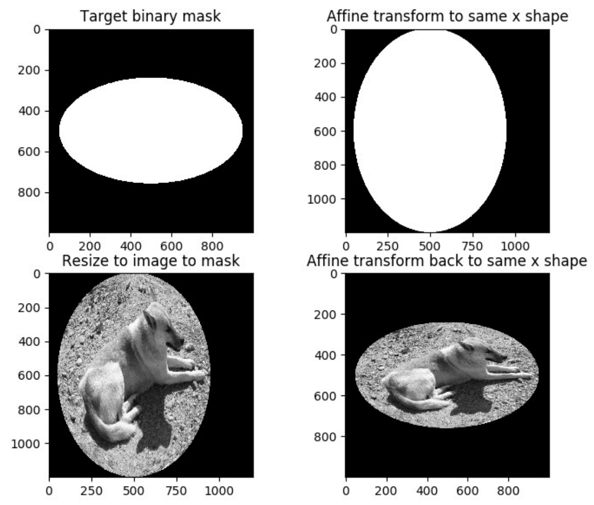
> Design a computer program for geometric transformation of an image. Try to find the optimal geometric transformation to obtain the warped images shown below. Describe your approach as clearly as possible and show the resulting images. You may also challenge yourself by designing an interactive interface for more flexible geometric transformation.

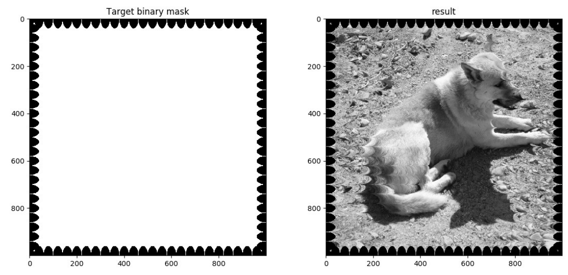
We can make full use on affine transform.

Here is my procedure

- Given a target binary mask
- Using affine transform to make the mask same as original image in x axis.
- Resize the image into mask row by row.
- Using the same affine metrix to make the image back to the original size of target mask.

And result





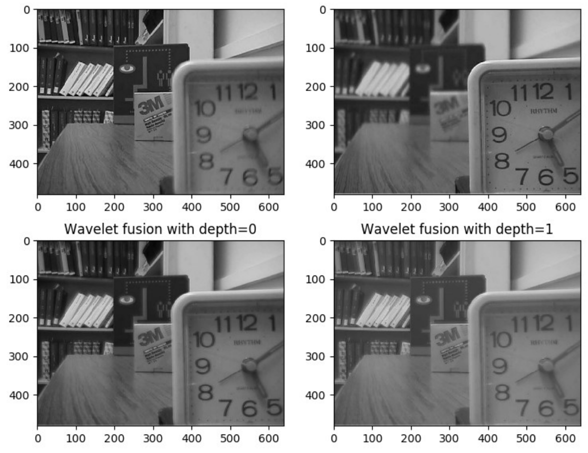
Part 2: (30%) Image Fusion

Using Wavelet Transform. The algorithm of image fusion using DWT is described in the following steps:

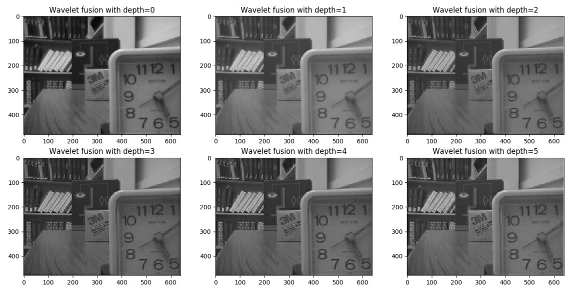
1. The size of images to be fused needs to be the same size and the resolution needs to be of power of two.
2. The two dimensional Discrete Wavelet Transform (DWT) should be applied to the resized images.
3. Fusion rule: The most used image fusion rule using wavelet transform is maximum selection, by comparing the DWT coefficients of the two (or more) images and select the maximum. While the lowpass subband is an approximation of the input image, the three detail subbands convey information about the detail parts in horizontal, vertical and diagonal directions. Different merging procedures will be applied to approximation and detail subbands. Lowpass subband will be merged using simple averaging operations since they both contain approximations of the source images, and the maximum selection rule is applied to detail subbands, as shown in the following figure.
4. After selecting the fused low frequency and high frequency bands, fused image is reconstructed using the inverse DWT from on the subbands determined in step 3. Implement an image fusion program applying the DWT method as described above.
5. Test your program with the image sets provided in this homework and image sets you wish to test.
6. Quantitatively and qualitatively compare and discuss the effect of the image fusion results using different

scales of decomposition.

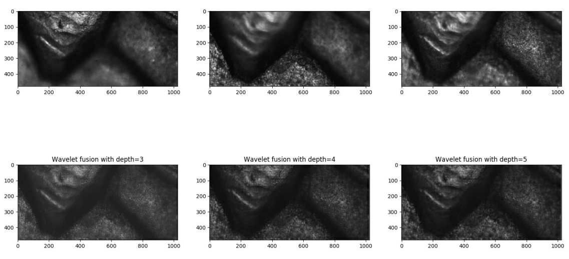
I compare the result to simply average two images together(depth=0).

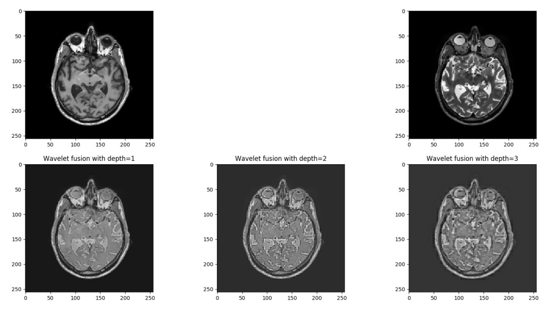


Also, I try average the low frequency data and find maximen in high frequency. When the depth is more deeper, the fusion effect will be more better. However, when it goes too deep, the image become course. Thus, I found the best depth is 2 or 3.


Wavelet fusion with depth=0
Wavelet fusion with depth=1
Wavelet fusion with depth=2
Wavelet fusion with depth=3
Wavelet fusion with depth=4
Wavelet fusion with depth=5

And try on other dataset, the best depth value for wavelet is case by case.


Wavelet fusion with depth=3
Wavelet fusion with depth=4
Wavelet fusion with depth=5

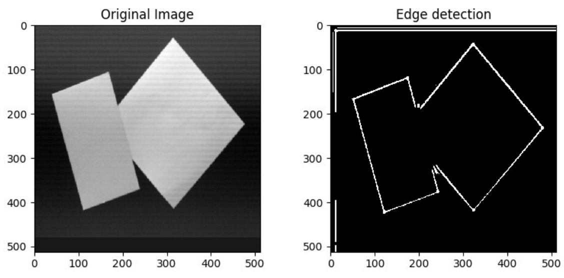

Wavelet fusion with depth=1
Wavelet fusion with depth=2
Wavelet fusion with depth=3

```
imgs = ["data/part2/set1/clock1.JPG", "data
imgs = ["data/part2/set2/multifocus1.JPG",
imgs = ["data/part2/set3/MRI1.jpg", "data/p
```

Part 3: (40%) Hough Transform

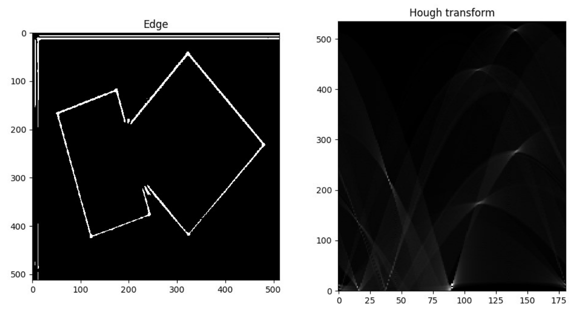
> Apply Hough transform method described in 10.2 of our textbook to find the sides of the two rectangular papers in the image RECTS.BMP (available from the CEIBA course website). Determine the areas and perimeters of the two rectangular papers assuming the scale factor is 0.5 mm/pixel in both horizontal and vertical directions. Try your Hough transform program for other images containing multiple lines. What are the major factors affecting the performance of your Hough transform program. Discuss and report your results in details.

First, using LoG(in hw3) to get where the edge is. I binarize the image by absolute value greater than 0.5. The result show in below.

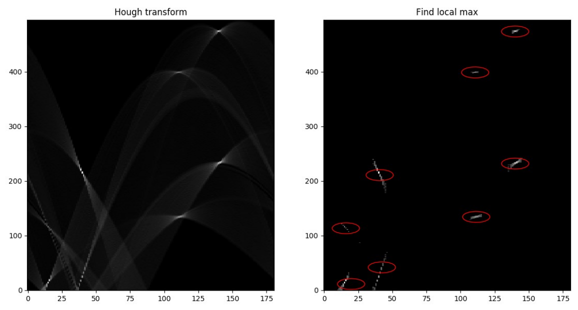


Then applying hough transform. Just calculate each curve by $x\cos(\theta) + y\sin(\theta) = r$

```
th = np.linspace(0, np.pi, 181)
x, y = np.where(edge)
r = np.int(np.outer(x, np.cos(th)) + np.out
hough = np.zeros([2 * max_dis, th.size])
for i in range(len(x)):
    hough[r[i] + max_dis, np.arange(th.size
```



I use kmean to find the local max. Note that rectangle has 4 edges, so the number of cluster is 8.

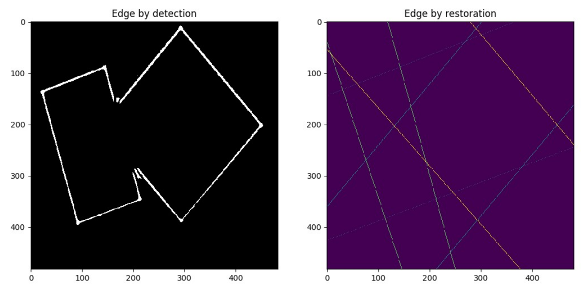


Here are center coordinates for each cluster.

```
[   21.46595995,  817.20094229],
[   20.69287289, 1081.7038942 ],
[   49.98192989, 1156.67437658],
[   50.02811562,  914.84441636],
[ 105.26107446,  798.21677663],
[ 108.29040657,  671.29641498],
[ 130.16452809,  897.08511548],
[ 131.4446664 ,  647.58409908],
```

We can pair each two cluster when its angle(First row) are similar. The same angle mean the line is parallel, which is agree with the property of rectangle.

We can see the evidence by restored back



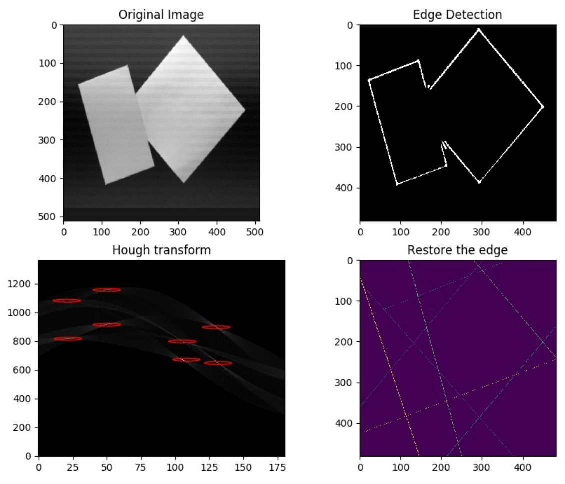
And calculate the distance in each pair, which is the width of square.

The distance of each pair is `264.50295191, 241.82996022, 126.92036165, 249.5010164`

Right rectangle is contructed by pair1 and pair3, so the area is `241 * 249 * 0.5 ** 2` mm^2

Left rectangle is constructed by pair0 and pair2, so the area is `264 * 126 * 0.5 ** 2` mm^2

Summary:



See `hw6/test_hough.py`

Program usage

Install on local machine

```
./setup.sh
```

Install by Docker

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

Without QT

You can use command line as a postfix image processor.(Same as HW2)

For example

- `python3 hw6_np.py --read data/part3/rects.bmp --log 3 --threshold 50 --hough --show-noasp`
- `python3 hw6_np.py --read data/part2/set1/clock1.JPG --graya --wavelet 1 --show`
- `python3 hw6_np.py --read data/part1/IP_dog.bmp --read data/mask.png --graya --threshold .5 --geo --show`
- `python3 hw6_np.py --read data/part2/set1/clock1.JPG --graya --read data/part2/set1/clock2.JPG --graya --fusion 3 --show`

You can use `python3 hw6_np.py --help` to see more.

QT

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

A dynamic UI image processor.

Note:

- All filter should run under gray scale image.
- The parameters of gaussian is cutoff value not sigma.
- To display the Difference operation in convenience, the code is not only `img1 - img2`. Instead, I use `(img1 - img2) / 2 + 127`.
- Add colormap creation module before using pseudo color module
-
- I using qt built-in `QColorDialog.getColor` as a color picker.

demo Image