Assignment 1

Problems

A. ‘i’ - reload the original image
B. ‘w’ - save the current image
C. ‘g’ - convert the image to gray scale
D. ‘G’ - covert the image to gray scale using my function
E. ‘c’ - cycle through the color channels
F. ‘s’ - smoothing using built-in functions
G. ‘S’ - smoothing using my functions
H. ‘d’ - downsample without smoothing
I. ‘D’ - downsample with smoothing
J. ‘x’ - convolution with x derivative
K. ‘y’ - convolution with y derivative
L. ‘m’ - show gradient magnitude
M. ‘p’ - plot gradient vector
N. ‘r’ - rotate image
O. ‘h’ - print helper

Solution

A. ‘i’ is pressed. Read original image and adjust size if too big. Copy the original image to processed image canceling previous processing.
B. ‘w’ is pressed. Save the current processed image to output image.
C. ‘g’ is pressed. Convert the image using built-in cvtColor function.
D. ‘G’ is pressed. Conver the image using the my function that g = 0.299x + 0.587y + 0.114z
E. ‘c’ is pressed. Use a counter “channel” from 0 to 2 to count the channel. Every time c is pressed, increase counter by one or let the counter equals 0 if counter is 2.
F. ‘s’ is pressed. Use medianBlur to smooth the image
G. ‘S’ is pressed. Create a smoothing kernel and pass it to myConvolve function to do the convolution. myConvolve function is written by cython and compiled to speed up the convolution. In myConvolve function, zero padding is used for corner pixels.
H. ‘d’ is pressed. Resize the image by 2.
I. ‘D’ is pressed. Downsample the image using gaussian pyramid.
J. ‘x’ is pressed. Calculate and normalize the x derivative using sobel convolution.
K. ‘y’ is pressed. Calculate and normalize the y derivative using sobel convolution.
L. ‘m’ is pressed. Calculate and normalize the x and y derivative using sobel convolution.
M. ‘p’ is pressed. Calculate the gradient vector for every N pixels. Convert to angle

coordinate. Find the ending point of the line. Draw the line on the image.
N. ‘r’ is pressed. Rotate the image.
O. ‘h’ is pressed. Print helper function

Implementation

A simple python program is provided as AS1.py. The program defines two global variable ‘img’ and ‘img_p’ to store original image and processed image. A while loop is implemented to provide image processing operations.
Different track bars are implemented to control the user defined variables.
To convert the image to gray scale using my function, the program uses the equation g = 0.299x + 0.587y + 0.114z.
A convolution function is implemented in cython and compiled to speed up the convolution. My convolution function uses zero padding for the corner pixels.
And compiled cython function is included in the submission, there’s no need to compile it using python setup.py build-ext --inplace
The program can import myFunction directly.
The program will read the input file ‘images.jpeg’
Output file is saved as ‘output.jpg’
To run the program, open a terminal and type “python AS1.py”
To test cython function, a test.py is provided. The program performs convolution using the same algorithm.

Results

A. ‘i’ is pressed. Read original image and adjust size if too big. Copy the original image to processed image canceling previous processing.

B. ‘w’ is pressed.
C. ‘g’ is pressed. Convert the image using built-in cvtColor function.

D. ‘G’ is pressed. Conver the image using the my function that g = 0.299x + 0.587y + 0.114z

E. ‘c’ is pressed. Use a counter “channel” from 0 to 2 to count the channel. Every time c is pressed, increase counter by one or let the counter equals 0 if counter is 2.

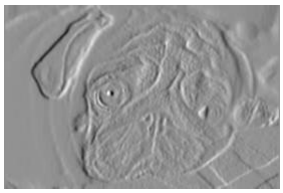
F. ‘s’ is pressed. Use medianBlur to smooth the image

G. ‘S’ is pressed. Create a smoothing kernel and pass it to myConvolve function to do the convolution. myConvolve function is written by cython and compiled to speed up the convolution. In myConvolve function, zero padding is used for corner pixels.

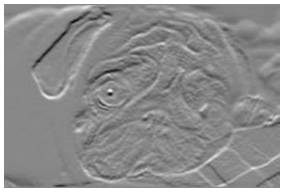
H. ‘d’ is pressed. Resize the image by 2.

I. ‘D’ is pressed. Downsample the image using gaussian pyramid.

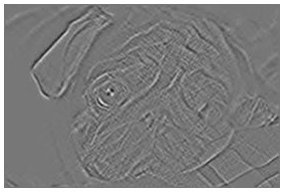
G. ‘x’ is pressed. Calculate and normalize the x derivative using sobel convolution.



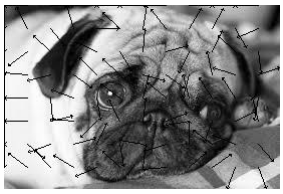
K. ‘y’ is pressed. Calculate and normalize the y derivative using sobel convolution.



L. ‘m’ is pressed. Calculate and normalize the x and y derivative using sobel convolution.



I. ‘p’ is pressed. Calculate the gradient vector for every N pixels. Convert to angle coordinate. Find the ending point of the line. Draw the line on the image.



J. ‘r’ is pressed. Rotate the image.

K. ‘h’ is pressed. Print helper function

```
i=reload image
w=save current processed image
g=convert to grayscale
G=convert to grayscale using my function
c=cycle through color channels
s=smoothing
S=smoothing usiong my convolution function
d=downsample by a factor of 2 without smoothing
D=downsample by a factor of 2 with smoothing
x=x derivative filter
y=y derivative filter
m=magnitude of gradient
p=plot gradient vector
r=rotate image
```

Testing performance

```
(base) yinjiang@dhcp253 AS1 % python3 test.py
--- Cython Function: 0.19259095191955566 seconds ---
--- Python Function: 0.2069101333618164 seconds ---
(base) yinjiang@dhcp253 AS1 %
```

The cython function is a little faster than the python function.