

1651030

TRẦN THANH TÂN

CS412 – HOMEWORK 01

I. INTRODUCTION

- A simple image manipulation which is implemented using Python and OpenCV library to solve the requirements.

II. PROJECT

1. Structure

- Document
 - 📄 Report.doc: Report file of the project
- Source Code:
 - 📄 Main.py: Source file
- Release:
 - 📄 Main.exe: Final release .exe file
 - 📄 Input.jpg: Default input image for the project to run

2. Requirement

| NAME | STATUS |
|---|--------|
| When capturing images from a camera allow the program to capture and process the images continuously. | Done |
| The main parameters of each algorithm should be made available for interactive manipulation through keyboard/mouse/trackbar interaction. | Done |
| Your program must include a help key describing its functionality. | Done |
| You need to evaluate the performance of the algorithm you choose using test data at least three images (e.g. adding noise or occlusions). The results of your evaluation should be included in your report. Try to determine the strengths and weaknesses of the algorithm. | Done |
| Your report must include a description of the algorithm you implemented | Done |
| Your submission must organize into 3 folder and compressed in 1 file StudentID.zip | Done |

3. Functions

| NAME | DESCRIPTION | USAGE |
|--------------------|--|---------------------------------------|
| def main(filename) | Main function of the program Decide to load the image or capture it through the camera by defining in line 218. Main('./input.jpg'): For loading image path. | Load process image / Capture image |

| | | |
|--|--|---|
| | Main(''): For capturing image using camera. | |
| def load_image(filename) | Loading the image using the file name. | Initialization of the program. Reload the program |
| def output_image(image) | Output the image that is passed in | Output the final 'output.jpg' file |
| def openCV_gray_scale(filename)(*) | Gray scale the image using OpenCV default function | Process grayscaled image |
| def algo_gray_scale(filename)(*) | Gray scale the image using the algorithm function | Process grayscaled image |
| def cycle_image(filename) | Cycle through the R, G, B channels of the image everytime users press 'c' key | Get image's R,G,B channels |
| def smooth_image(filename)(*) | Blur the image using the OpenCV default function which can control by trackbar on the window | Process blurred image |
| def algo_smooth_image(filename) (*) | Blur the image using the implemented function which can control by trackbar on the window | Process blurred image |
| def convolution_x_derivative(filename) | Convert the image to grayscale and perform convolution with an x derivative filter. Normalize the obtained values to the range [0,255]. | Show the x image derivative |
| def convolution_y_derivative(filename) | Convert the image to grayscale and perform convolution with an y derivative filter. Normalize the obtained values to the range [0,255]. | Show the y image derivative |
| def rotate_Q_angle(filename) | Convert the image to grayscale and rotate it using an angle of Q degrees. Use a track bar to control the rotation angle. | Rotate the image |
| def magnitude(filename) | Show the magnitude of the gradient normalized to the range [0,255] | Show the image's magnitude |
| def help() | Show help keys | Show help keys |









4. Procedure steps

1. Run the **main.exe** file.
2. Wait for the Image window to appear . (This may take a few seconds)
3. By default, it'll generate the **logo.jpg** image in the folder. If you want to change the input, you can copy your image into and change the name as '**logo.jpg**' and run the program again.
4. If you want to run the program directly, run the **main.py** file in the Sourcode folder by using the **python main.py** command and follow the instruction of the main function above.

Note: You can only perform one process type and then output the image or load the original image again, then press any keys to close the window. Or just press "i" key to reload the original image then move to any next key to process.

III. EVALUATION













1. Gray scale Image

| ORIGINAL | OPENCV | MY ALGORITHM |
|--|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

Comments:

1. The grayscale function of OpenCV can output the better result than my algorithm.
2. With my algorithm, I just calculated the gray index = $(\text{blue} + \text{green} + \text{red}) / ((\text{blue} + \text{green} + \text{red}) / 3)$
3. The OpenCV function work better with normal image (without noise)
4. The OpenCV function is more suitable for situation that need 80% accuracy to process
5. With my algorithm, the processed image seems to be more detail. Especially with noise image, as we can see, the bird can be looked clearer or the shadow of the third image is easier to recognized than the OpenCV's.


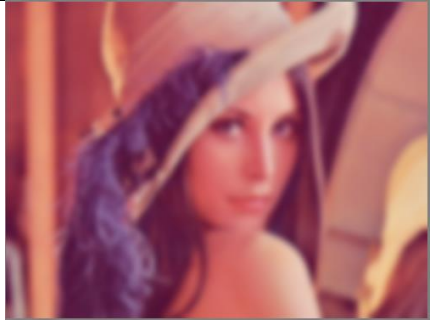






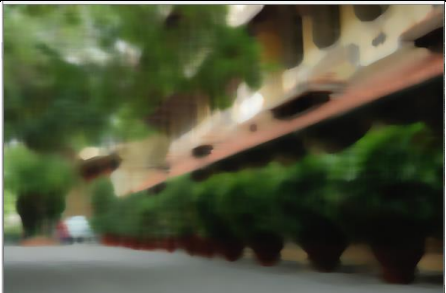
2. Cycle channels

| ORIGINAL | OPENCV | MY ALGORITHM |
|--|---|---|
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Comments:

1. The output of three channels from three images are 80% similar.
2. The noise images can be easily identified by looking at the Red channel which result the noise more than the original one.
3. The Green and Blue channel looks the same for all images.







3. Smooth image

| ORIGINAL | OPENCV (Gaussian) | MY ALGORITHM (Median) |
|---|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

Comments:

1. By applying different algorithm of blurring image (Gaussian and Median) with the same index controlled by the trackbar. The result was when I scrolled the OpenCV gaussian function to the maximum index (100), but only ≤ 10 for Median algorithm.
2. It looks like the OpenCV function Gaussian can process an image looking smoother than the Median.
3. The processed image using the Median is not as clear as the OpenCV's, but it's more effective for completely blurring image (if you need to process an image that can't be recognized from the original one \rightarrow should use this)

4. Convolution X and Y – Magnitude

| ORIGINAL | X | Y |
|--|---|--|
|  |  |  |
|  |  |  |
|  |  |  |