

cs512 - Computer Vision – Fall20 - Assignment 1: Report

Yash Patel, A20451170

Department of Computer Science

Illinois Institute of Technology

October 8, 2020

- **Abstract**

This is a report for programming assignment 1. It covers the basic image manipulation using OpenCV and other libraries.

- **Problem Statement**

The program should load an image by either reading it from a file or capturing it directly from a camera. When the user presses a key perform the operation corresponding to the key on the original image (not the result of the last processing step). The program should satisfy the following specifications:

- 1] image should be either read from a file or captured directly from a camera. When capturing an image from the camera, continue to capture and process image continuously.
- 2] the image should be read as a 3-channel color image.
- 3] The program should work for any size image.

- **Environment**

Python version: 3.6 (anaconda)

Libraries:

- 1) OpenCV-python version: 4.4.0.44
- 2) Convolution (cython code)

- **Implementation Details:**

Special keys on the keyboard should be used to modify the displayed image as follows:

- a) **'i'** – Reload the original image. (i.e. cancel any previous processing)
 - a. When key is pressed application will be switched to process image input.
- b) **'l'** – capture the image from camera.
 - a. when key is pressed application will be switched to process camera input.
- c) **'w'** – Save the current (possibly processed) image into the file 'out.jpg'.

- d) **'g'** – Convert the image to grayscale using the OpenCV conversion function.
 - a. Used OpenCV `cvtColor` method to convert color image to grayscale image.
- e) **'G'** – Convert the image to grayscale using your implementation of conversion function.
 - a. First used average of all channels. But output was different than OpenCV's output.
 - b. Then used 0.299 ratio for red channel, 0.587 for green channel, 0.114 for blue channel which was very similar to OpenCV's.
- f) **'c'** – Cycle through the color channels of the image showing a different channel every time the key is pressed.
- g) **'s'** – Convert the image to grayscale and smooth it using the OpenCV function. Use a track bar to control the amount of smoothing.
 - a. For smoothing, first used average filter but output of this filter was not satisfactory for large amount of smoothing. Then used gaussian filter which was very satisfactory on large amount of smoothing.
 - b. For gaussian smoothing I used OpenCV's `GaussianBlur` method which takes smoothing amount as parameters and standard deviation in x and y directions.
- h) **'S'** – Convert the image to grayscale and smooth it using your function which should perform convolution with a suitable filter.
 - a. Here I used gaussian filter same as before.
 - b. First, I wrote code in python. But execution was very slow. Then I wrote code in cython which was also slow but fast comparing to the code in python.
- i) **'d'** – Down sample the image by a factor of 2 without smoothing.
 - a. Here I used OpenCV's `resize` method.
- j) **'D'** – Down sample the image by a factor of 2 with smoothing.
 - a. First, I applied gaussian smoothing on the image.
 - b. Then down sampled using OpenCV's `resize` method.
- k) **'x'** – Convert the image to grayscale and perform convolution with an x derivative filter. Normalize the obtained values to the range [0,255].
 - a. To get x derivative filter I used OpenCV's `getDerivKernels` which returns two 2D column matrix x [-1,0,1] and y [1,2,1] values of sobel x derivative filter. Then multiplied both matrix and got x derivative of sobel filter.
 - b. After making a filter I used OpenCV's `filter2D` method to convolve an image with this filter.
 - c. First, I applied two 1D convolution but output was not good enough then I applied one 2D convolution which shows better performance to find vertical edges.
- l) **'y'** – Convert the image to grayscale and perform convolution with an y derivative filter. Normalize the obtained values to the range [0,255].
 - a. To get y derivative filter I used OpenCV's `getDerivKernels` which returns two 2D column matrix x [1,2,1] and y [-1,0,1] values of sobel derivative filter. Then multiplied both matrix and got y derivative of sobel filter.
 - b. Further process is same as before.
- m) **'m'** – Show the magnitude of the gradient normalized to the range [0,255].
 - a. First, I got x and y derivatives of the image.
 - b. To find magnitude I used NumPy's `hypot` method which gives Euclidean distance.

- n) 'p' – Convert the image to grayscale and plot the gradient vectors of the image for every N pixel and let the plotted gradient vectors have a length of K. use a track bar to control N. plot the vectors as short line segments of length K.
 - a. Using x and y derivatives and magnitude of the gradients I found angle of the vector
 - b. Then I used OpenCV's arrowedLine to draw a vector on the image which takes start pixel, end pixel, color, thickness of line as parameters.
- o) 'r' – Convert the image to grayscale and rotate it using an angle of theta degrees. Use a track bar to control the rotation angle. The rotation of the image should be performed using an inverse map so there are no holes in it.
 - a. First, I found rotation matrix using OpenCV's getRotationMatrix2D
 - b. Second, I found new boundaries of the image.
 - c. Then apply warpAffine method to rotate image without holes in rotated image.
- p) 'h' – Display a short description of the program and the keys it supports.

• Result and Discussion

Here I used a colorful parrot's image to observe the outputs of different processing algorithms.





1) Grayscale image using OpenCV and own implementation:

To get a grayscale image I used COLOR_BGR2GRAY parameter in OpenCV's method and weight [0.299, 0.587, 0.144] for [r, g, b] in my own implementation. Here I can see that both implementations give similar result.

OpenCV conversion	Own conversion
-------------------	----------------







2) Cycle through the color channels:

Red channel		
		
Green channel		
		
Blue channel		



3) Smoothing using OpenCV and own cython implementation

OpenCV	Cython
Smoothing amount 9	Smoothing amount 9
	
Smoothing amount 5	Smoothing amount 5
	

Smoothing amount is controlled by track bar in the application. Here I took two different smoothing amounts (5 and 9) for OpenCV and cython implementation. Here I used 1 to 31 as a range for smoothing amount.

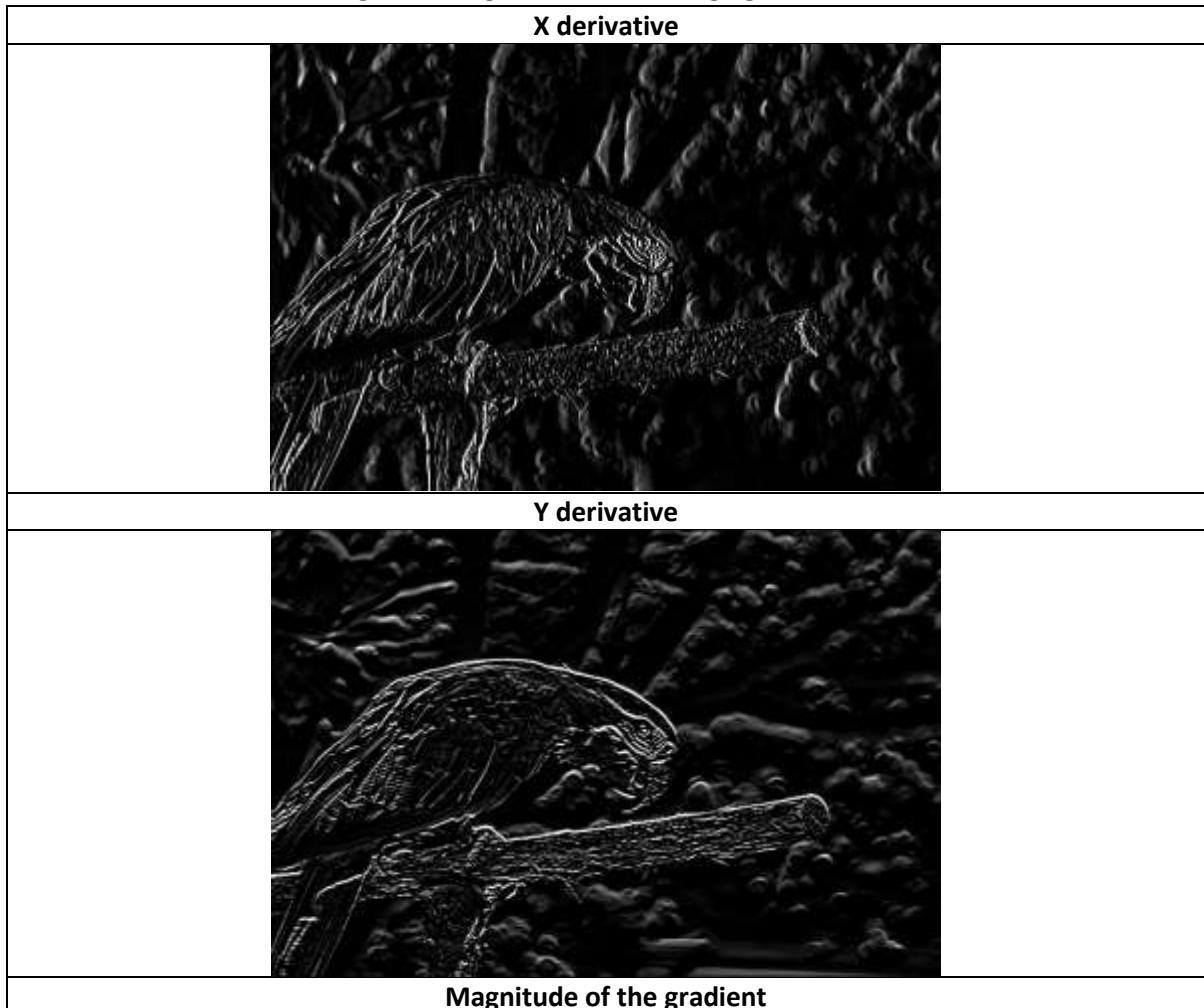
Here smoothing amount is a size of gaussian filter. I can see that as we increase smoothing amount image gets more blurred in both implementations. Also, I can see a dark border around the image as we increase smoothing amount.

4) Down sample image by factor of 2 with and without smoothing



In with smoothing I first applied gaussian filter with smoothing amount of 31 and then down sampled the image by factor of 2. Here I can observe that after smoothing and resizing, an image looks more blurred than smoothed image without resizing. It shows the information loss in the image.

5) X and Y derivative of the image and magnitude of the image gradient





For x derivative sobel filter I used derivative in x direction and smoothing in y direction.

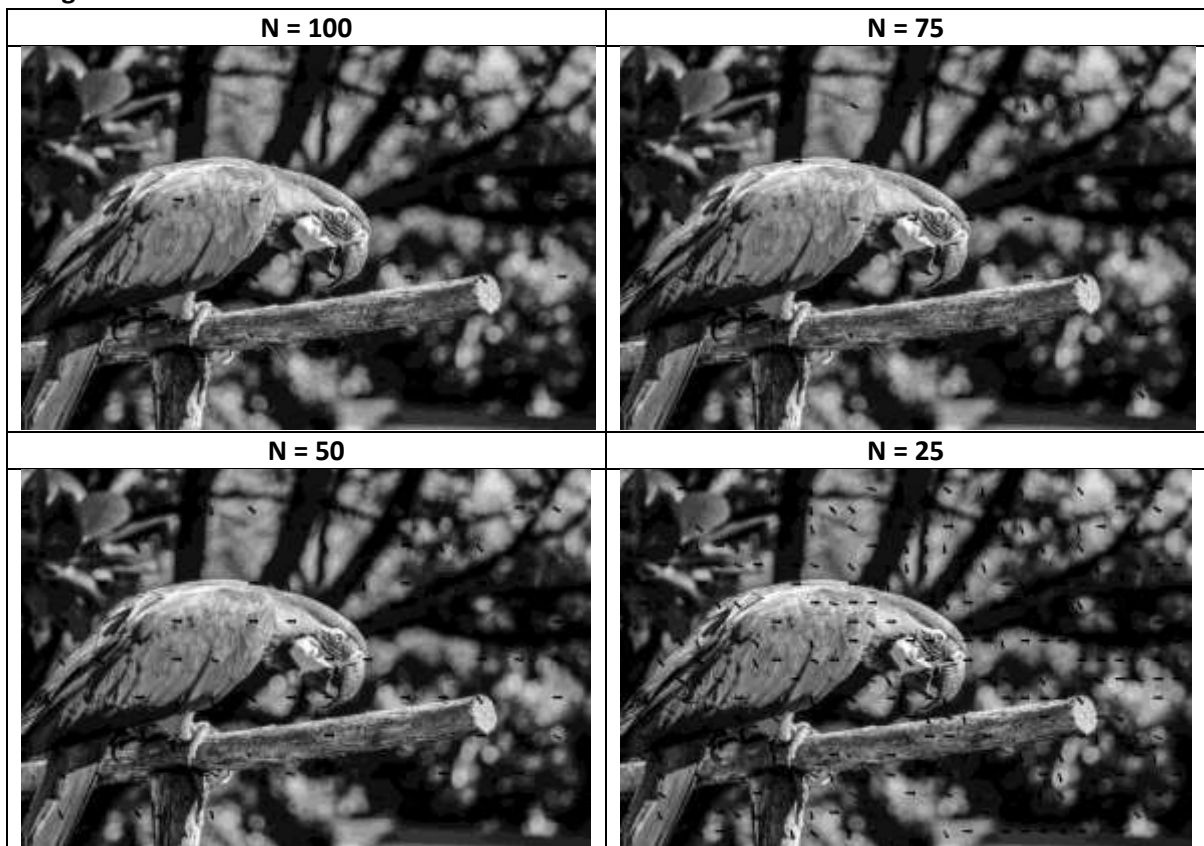
For y derivative sobel filter I used derivative in y direction and smoothing in x direction.

In x derivative image I applied x derivative sobel filter which gave me vertical edges in the image and smoothed image and horizontal direction.

In y derivative image I applied y derivative sobel filter which gave me horizontal edges in the image and smoothed image and horizontal direction.

In magnitude of the gradient I can see both vertical and horizontal edges in the image and smoothed image in both directions.

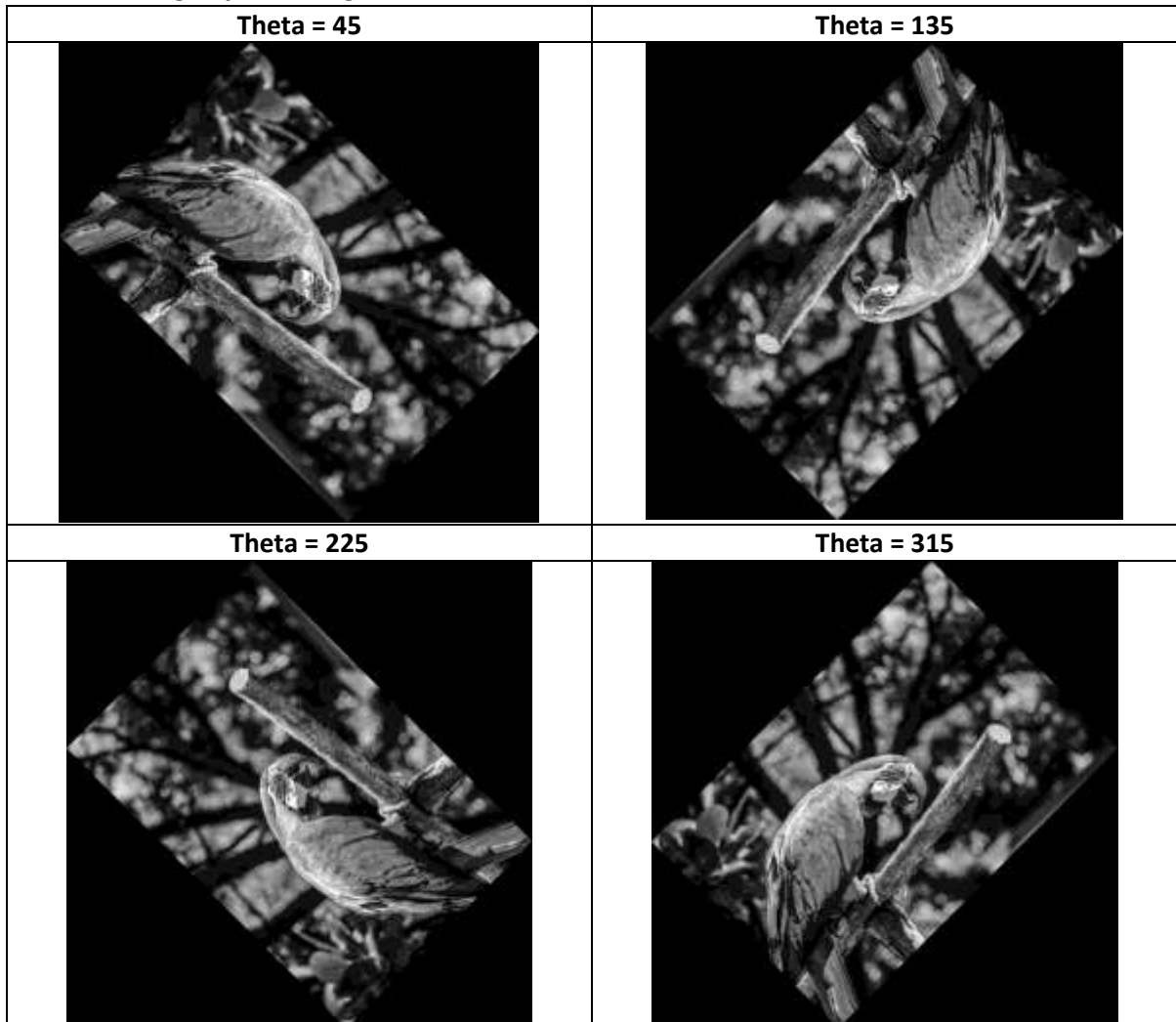
6) Plot gradient vectors



I took 10 units as a gradient vectors' size. As I decreased the value of N I got dense gradient vectors.

I can see that gradient vectors have different directions here which shows the direction of change at that pixel in the image. Here I used 100 as a maximum range for N.

7) Rotate an image by theta degree



Here I experimented four different angle values (45, 135, 225, 315) that I changed with track bar. Here I can see that when warpAffine is applied on the image, it rotates an image with rotation matrix and apply new boundaries to an image.

- **References**

- 1) <https://opencv-python-tutroals.readthedocs.io/en/latest/index.html>
- 2) <https://cython.readthedocs.io/en/latest/src/tutorial/numpy.html>