cs512 Assignment 1

Chandni Patel
Department of Computer Science
Illinois Institute of Technology

October 8, 2020

Abstract

This report contains description of the given problem, the algorithms used to solve the problem, and analysis of the results obtained for the programming part of Assignment 2.

1 Problem Statement

The goal is to write a program to perform simple image manipulation using OpenCV with following requirements:

- The program should load an image by reading it from a file or capturing it directly.
- The image should be read as a 3 channel color image.
- The program should work on any size of image.
- On pressing a support key, apply corresponding function on the original image.
- When implementing convolution use Cython to speed up execution.

2 Proposed Solution

Solution for this programming assignment in done using Python and Cython. OpenCV and custom functions are used to achieve various required functionality.

3 Implementation Details

Most of the functions were easy to implement using the OpenCV documentation. I have explained in detail how I executed each function in the next section.

One of the major issue was to setup Cython as it took a long time. The convolution itself is very slow for smoothing especially when we take higher values. Initially, it crashed Jupyter notebook and had to restart the kernel again. Now, it is at a more stable state.

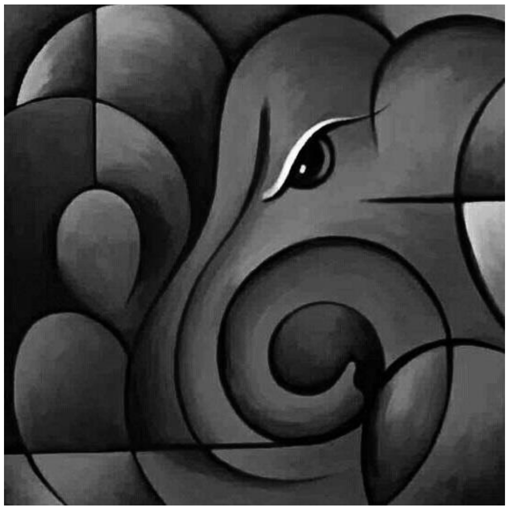
Instructions to run the program:

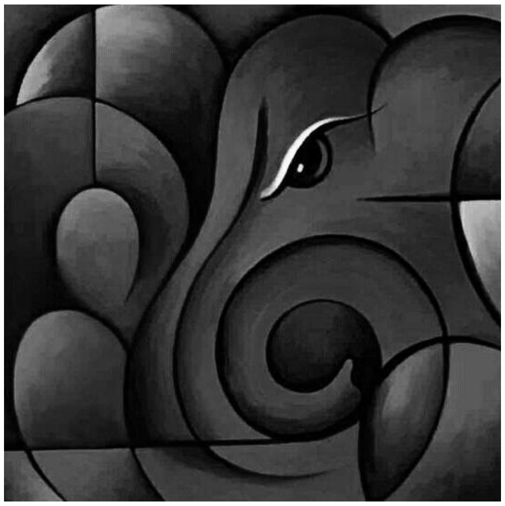
- I have added default input image name in the code for which the Jupyter notebook and in.jpg image will have to be in the same folder location.
- To test the camera capturing feature, replace the 'in.jpg' parameter with '' empty string where in class instance is being created (last line of code).
- Run src\Cython\Setup_file.py for import smoothing_convolution in code to work.

4 Results and Discussion

Here are the results obtained using the in.jpg image submitted along with the assignment:

1. **Grayscale image with OpenCV and conversion function:**
    - For OpenCV, the grayscale image is obtained using the cv2.COLOR_BGR2GRAY parameter in cv2.cvtColor function.
    - For conversion filter, it is obtained using the weights [0.299R, 0.587G, 0.114B].
    - On evaluation, both the images are correct, and functionality of both algorithms is efficient.


OpenCV


Conversion Filter

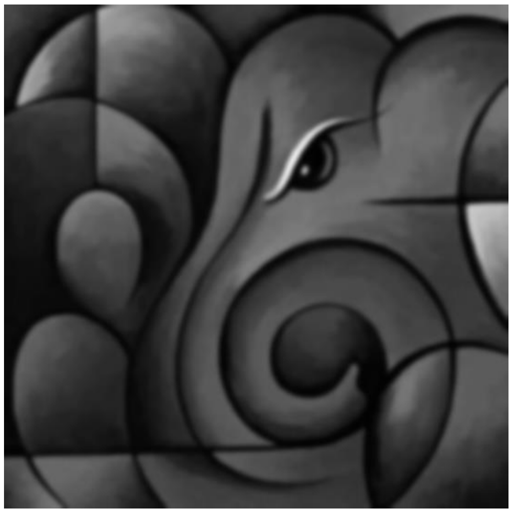
2. **Cycle through color channels of the image:**
    - Each channel is displayed one by one on pressing "c" again as follows:

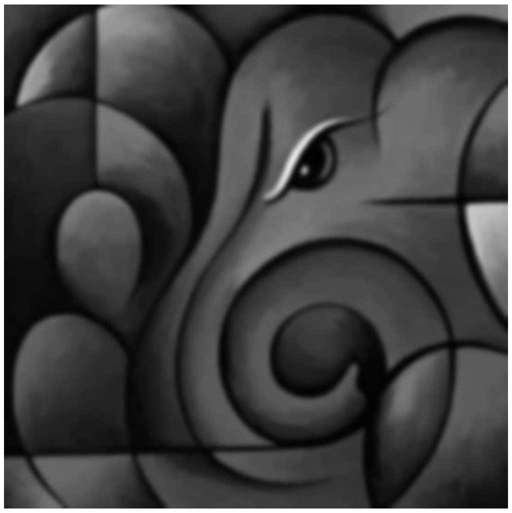
Red Channel

Green Channel

Blue Channel

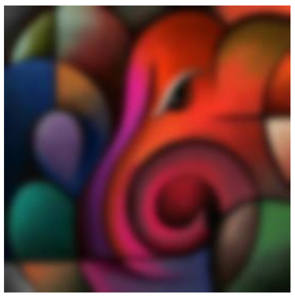
3. **Smoothing image with OpenCV and convolution function:**
   - For grayscale image, OpenCV is used as before.
   - In OpenCV, the smoothing image is obtained using the (n,n) parameter in cv2.GaussianBlur function where n is obtained from the track bar to create the n*n filter.
   - In convolution filter, it is obtained using the Cython function by applying 1D filter in x direction first, and then applying 1D filter in y direction.
   - On evaluation, both the images are correct. But functionality of both algorithms is different. OpenCV is much faster, than the Cython function for convolution.
   - Here, n = 15 for both examples.


OpenCV

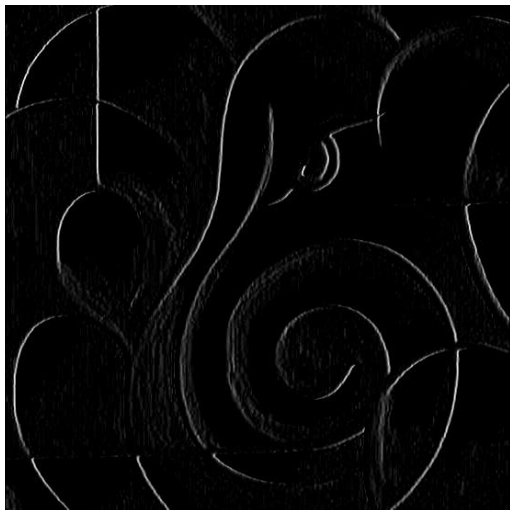

Cython function

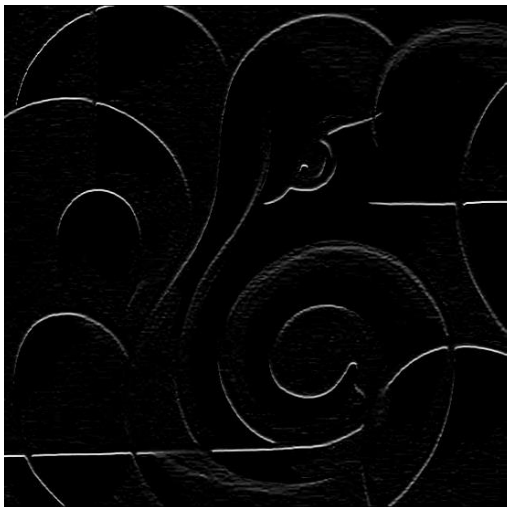
4. **Downsampling with and without smoothing:**
   - For without smoothing, the grayscale image is obtained using cv2.resize function with reduced dimensions as parameter.
   - For with smoothing, first smoothing is done using OpenCV as before and then using cv2.resize function with reduced dimensions as parameter.
   - On evaluation, both the images are downsized by 2.

Downsample without smoothing

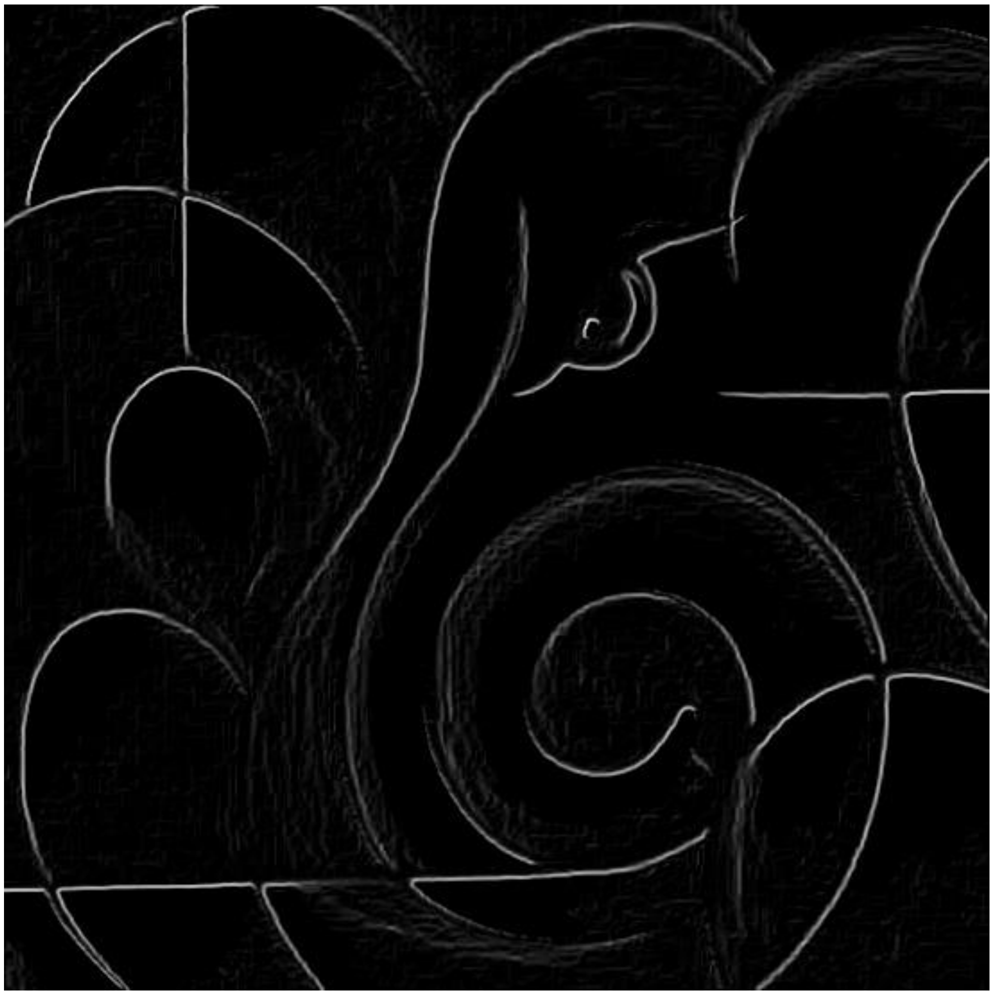

Downsample with smoothing

5. **X and Y derivative of the image:**
   - For grayscale image, OpenCV is used as before.
   - In both cases, cv2.getDerivKernels is used first to get k_x and k_y Sobel filters.
   - For x derivative, it is obtained by applying the cv2.filter2D function with k_y parameter for smoothing in y direction, then cv2.filter2D function with k_x.T parameter for derivative in x direction, and finally normalizing to obtain values in range of [0,255].
   - For y derivative, it is obtained by applying the cv2.filter2D function with k_x.T parameter for smoothing in x direction, then cv2.filter2D function with k_y parameter for derivative in y direction, and finally normalizing to obtain values in range of [0,255].
   - Here, x derivative shows all vertical edges and y derivative shows all horizontal edge present in the image.


X derivative


Y derivative

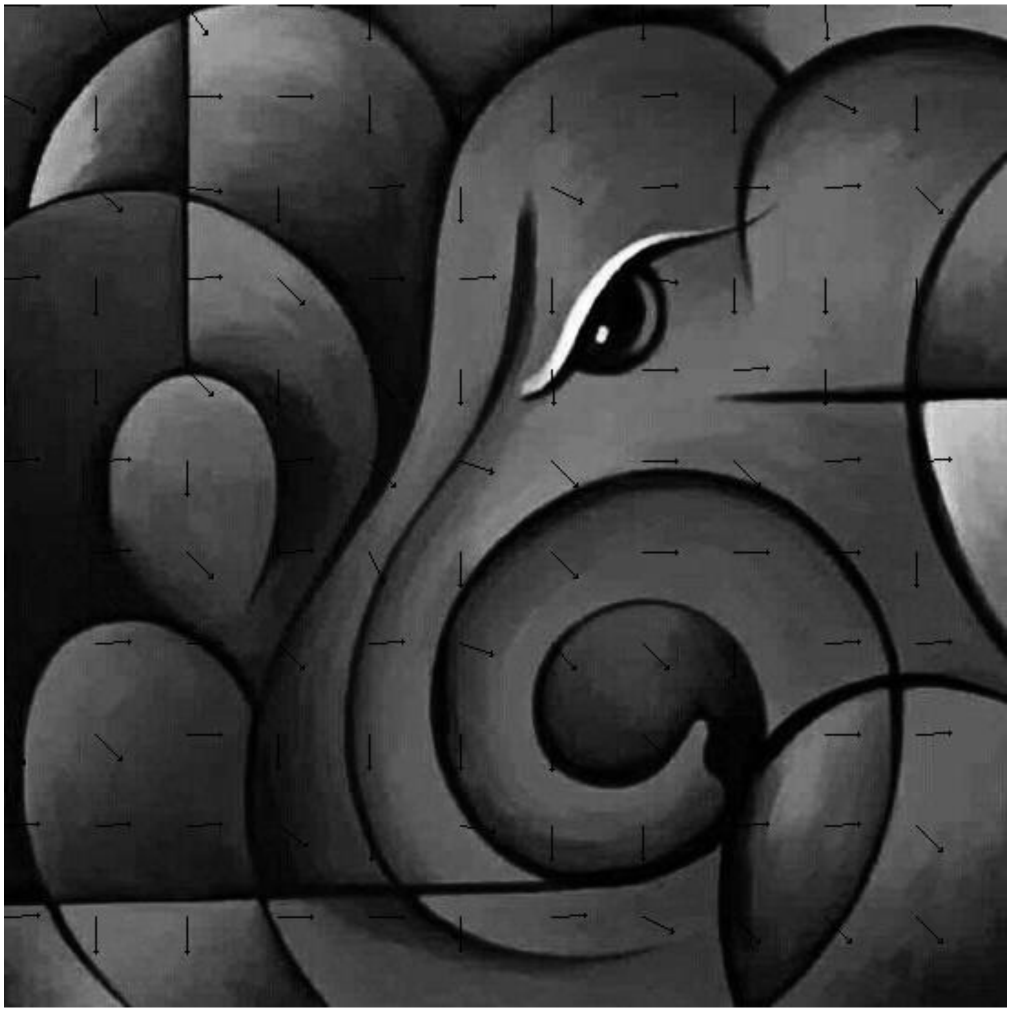
6. **Magnitude of gradient:**
    - For grayscale image, OpenCV is used as before.
    - Then x derivative and y derivative are obtained just as before.
    - For magnitudes of gradient, it is obtained using numpy.hypot function with gradient x and gradient y as parameters to get the square root of sum of gradient squares, and finally normalizing to obtain values in range of [0,255].



Magnitude of gradient

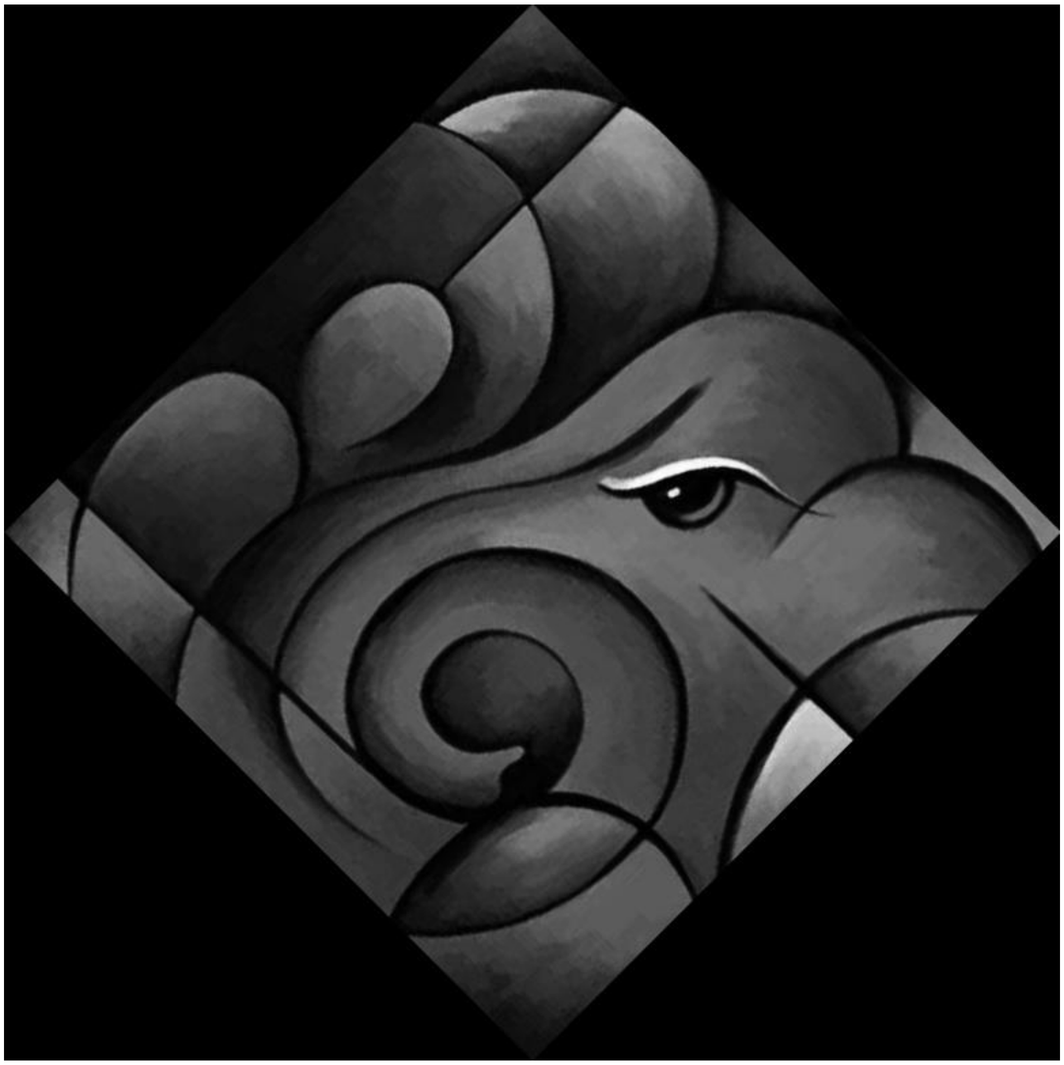
7. **Plot gradient vectors of image every N pixels:**
    - For grayscale image, OpenCV is used as before.
    - For plotting gradient vector, it is obtained by iterating through all the gradients and computing the vectors manually moving to the next N^th pixel where N is obtained from the track bar.

- Here, N is equal to 50 pixels.



Plotted gradient vectors

8. **Roatation of image:**
    - For grayscale image, OpenCV is used as before.
    - For rotation, it is obtained using cv2.warpAffine function. For clockwise rotation, taking negative theta value and theta is obtained from the track bar.
    - Here, width of new image is obtained by (number of rows * sin(theta)) + (number of columns * cos(theta)) and height is obtained by (number of rows * cos(theta)) + (number of columns * sin(theta)).
    - Here, theta is equal to 45 degrees.



Rotated Image

5 References

For some code reference to work with OpenCV: https://stackoverflow.com/