Assignment 2 Report

Course Code: CS 512
Course Name: Computer Vision
Course Instructor: Dr. Gady Agam

AS2
Prepared by: Khalid Saifullah
Student ID: A20423546
Date: 20th February, 2023

1. Problem Statement

The objective is to create a program that carries out basic image editing operations using OpenCV, while meeting the following conditions:

- The software must be able to read an image either from a file or from a direct capture.
- The image must be loaded as a color image with three channels.
- The software must be able to process images of any size.
- For faster execution of convolution, Cython should be used.

2. Proposed Solution

Python and Cython were utilized to develop a solution for this programming assignment. The required features were achieved by leveraging OpenCV and custom functions.

3. Implementation Details

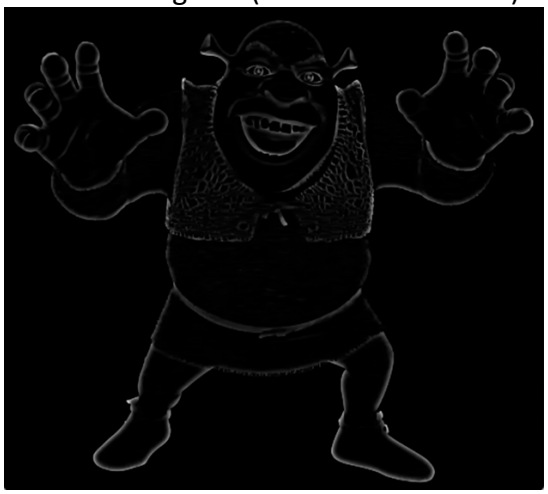
This rgb to grayscale conversion is done. It is simply done using the function “cv2.filter2D” and a gaussian smoothing kernel of 1 standard deviation. The trackbar attached is used to control the number of times the gaussian kernel is convolved with the original image.

Next, the same thing is done as the previous step but this time cython is used to convolve the gaussian filter and the image instead of using any builtin functions. Cython speeds up the process significantly as opposed to double loops as seen from the previous assignment.

Next, the original image is smoothed and downsampled by a factor of 2. The image size became from (892, 1000) to (446, 500). The function “cv2.resize” is used.

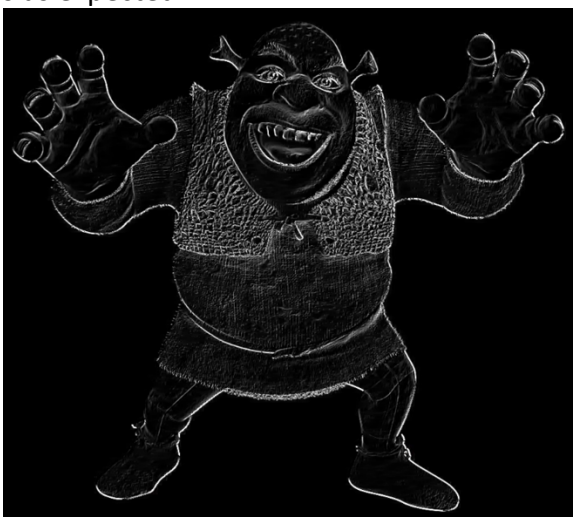
Now, the image from the previous step is upsampled by a factor of 2 and smoothed. The image size became from (446, 500) to (892, 1000). The function “cv2.resize” is used. It can be seen that the some contrast and details is lost from the cartoon’s clothes and the edges.

The difference between the two images is (after normalization):



The x-derivative and y-derivative of the images are given below. It can be seen that in the x-derivative image, changes in the y-direction is picked up by the gradients and vice versa. The gradients are computed using the function “cv2.sobel”.

Then the magnitude of the image gradient is computed. It captures both the x-derivative and y-derivative details as expected.

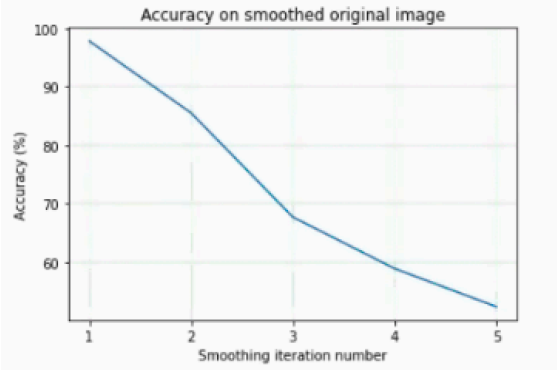


Then the gradient vectors were plotted on top of the original image every N pixels. A trackbar is also incorporated to change the value of N.

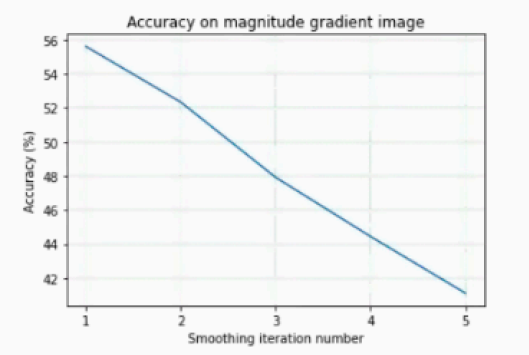
Then, the corners are detected using openCV's builtin function "cv2.cornerHarris". A trackbar is attached to change the threshold of the detected corners.

Then, the cornerness measure is computed using the correlation matrix and their eigenvalues. From the figure, very small red dots can be seen, although hard to see without zooming in. One reason for this could be not normalizing the values before plotting.

Next, a simple CNN network is used to perform MNIST digit classification. The images are smoothed before feeding into the network. The accuracy drops with the number of times we smooth the images. This is because, the edges are blurred and it becomes harder for the network to learn the features from the different digits.



Then, the magnitude of the gradient image of the smoothed images are fed into the network. This decreased the accuracy of the model even more. We can see that the highest accuracy is around 55% as opposed to 97% in the last step. The model can now extract lesser features because the smoothed images now have boundaries/edges that are even harder to detect by the gradients.



4. Results and Discussion

The cython html file gives an idea of which parts are using python. It looks like, our memory allocation and overall codes are just as efficient.

Generated by Cython 0.29.21

Yellow lines hint at Python interaction.
Click on a line that starts with a "+" to see the C code that Cython generated for it.

Raw output: cython_smoothing_ks.c

```
+01: import numpy as np
+02: import math
+03: import time
 04: # cimport numpy as np
 05:
+06: def ks_apply_convolution( int[:,:] in_img, int[:,:] filter_2d):
 07:
 08:     # cdef int[:,:] img = in_img
+09:     cdef int[:,:] out_img = np.zeros( (in_img.shape[0],in_img.shape[1]), dtype=np.intc)
 10:     cdef int x_img, y_img, x_filter, y_filter, img_x, img_y, x_fit, y_fit
+11:     cdef int pixel = 0
+12:     for x_img in range(0, in_img.shape[0], 1):
+13:         for y_img in range(0, in_img.shape[1], 1):
+14:             pixel = 0
+15:             for x_filter in range(-(filter_2d.shape[0]//2), (filter_2d.shape[0]//2)+1, 1):
+16:                 for y_filter in range(-(filter_2d.shape[1]//2), (filter_2d.shape[1]//2)+1, 1):
+17:                     img_x = x_img + x_filter
+18:                     img_y = y_img + y_filter
+19:                     x_fit = x_filter + (filter_2d.shape[0]//2)
+20:                     y_fit = y_filter + (filter_2d.shape[1]//2)
+21:                     if img_x < 0 or img_y < 0 or img_x >= in_img.shape[0] or img_y >= in_img.shape[1]:
+22:                         continue
+23:                     pixel += (filter_2d[x_fit][y_fit] * in_img[img_x][img_y])
+24:             out_img[x_img][y_img] = pixel
+25:     return out_img
```

5. References

1. https://cython.readthedocs.io/en/latest/src/quickstart/install.html
2. https://www.geeksforgeeks.org/how-to-install-opencv-4-on-macos/
3. https://www.geeksforgeeks.org/python-grayscaling-of-images-using-opencv/
4. https://stackoverflow.com/questions/14494101/using-other-keys-for-the-waitkey-function-of-opencv
5. https://answers.opencv.org/question/11959/program-termination/
6. https://www.geeksforgeeks.org/python-opencv-waitkey-function/
7. https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/