Cp467 A1 report

By Phoebe Schulman, Afaq Shad, Arvind Sahota

Abstract

In this project, the aim was to write a Python program to extract edges from the image. What we did was edge detection, using the Sobel edge detection algorithm.

Introduction

An edge can be thought of as the boundary of an object. Edge detection is used for processing images and can be done by checking for sharp changes in brightness. For example, a sudden change of black to white can imply that there's an edge of an object.

The concept we used is the Sobel edge detection algorithm. This uses two 3x3 gradient operators/kernels/filters. "The X kernel, when convolved with images, detects the horizontal edges, whereas the Y kernel detects vertical edges". Then, they're "combined together to form a new image which represents the sum of the X and Y edges of the image." (2)

Developed system

How our program works is by using Python API's including Matplotlib and Numpy.

```
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np
```

1. First, we get the image and use it as an array of shape (nx, ny, nz).

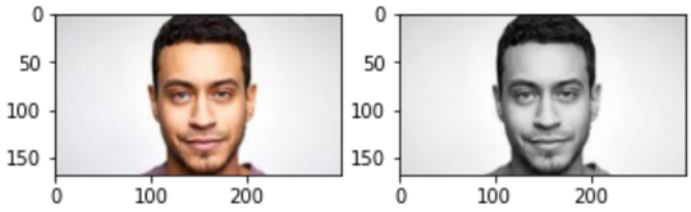
```
#import image
image_name = "face.jpg"
image_input = imread(image_name)
```

2. We convert the RGB image into a grayscale image, since colour doesn't have much meaning in finding an edge. This uses a dot product of the image and the vector [1,1,1]. *(2)*

```
#convert color into gray scale
grayscale_image = np.dot(image_input,[1,1,1])//3
grayscale_image = grayscale_image/255
```

So far, we have the original image and the gray image. We can display them. *(1)*

```
#display orig and gray image
fig1 = plt.figure(1)
ax1, ax2 = fig1.add_subplot(121), fig1.add_subplot(122) #2 grids of 1 by 1
ax1.imshow(image_input) #show orig image
ax2.imshow(grayscale_image, cmap=plt.get_cmap('gray')) #show gray second
fig1.show()
```



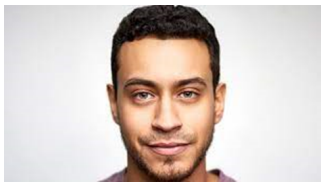
3. Now comes the edge detection by applying the Sobel operator. We define the gradients, GX and GY, as a 2D array. Then we traverse the pixels of the gray scale image, using the gradients to compute the edges in both the vertical and horizontal direction. It also uses the equation of gradient vector= sqrt (gx^2 + gy^2). *(1)*

```
#using sobel


#matrices associated with the Sobel filter
Gx = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
Gy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
"""
     _          _                  _          _
    |            |                |            |
    | 1   0  -1  |                |  1   2   1 |
Gx = | 2   0  -2  |    and     Gy = |  0   0   0 |
    | 1   0  -1  |                | -1  -2  -1 |
    |_          _|                |_          _|
"""
```

```
[rows,cols] = np.shape(grayscale_image) #gray image shape
sobel_image =  np.zeros(shape=(rows, cols))  # output image array (all elements are 0)


#traverses pixles - x and y direction #compute output
for i in range(rows - 2):
    for j in range(cols - 2):
        gx = np.sum(np.multiply(Gx, grayscale_image[i:i + 3, j:j + 3]))  # x direction
        gy = np.sum(np.multiply(Gy, grayscale_image[i:i + 3, j:j + 3]))  # y direction
        sobel_image[i + 1, j + 1] = np.sqrt(gx ** 2 + gy ** 2)  # calculate the "hypotenuse"
```

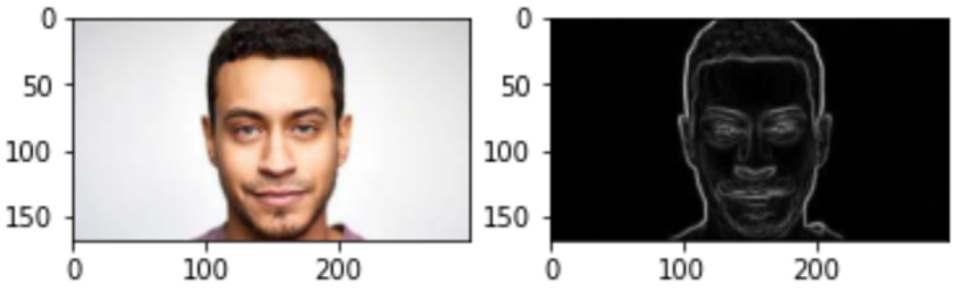
Results

Our input was a random image.



Our output is only the edges of the original picture. We compare and display the original and output image here.

```
#display orig and sobel image
fig2 = plt.figure(2)
ax1, ax2 = fig2.add_subplot(121), fig2.add_subplot(122)
ax1.imshow(image_input)
ax2.imshow(sobel_image, cmap=plt.get_cmap('gray'))
fig2.show()
```



Sources

1. Sobel edge detection from scratch using Python by Vedant Keshav Jadhav: https://coderspacket.com/sobel-edge-detection-from-scratch-using-python

2. Sobel-filter-tutorial by Adamiao: https://github.com/adamiao/sobel-filter-tutorial/blob/master/sobel_from_scratch.py