CS101 Project: Edge Detection Using Sobel Filtering

Faculty in Charge: Vipin K

Objective

The objective of this project is to implement Sobel filtering using Python to a given image for finding its edges. In image processing, an edge is the points where the image brightness changes sharply, i.e. the outlines. This is illustrated in the figures given below.

Fig.1 Input Image

Fig2. Image after edge detection

Input Considerations

You can assume that your input will be always a colour image in bitmap (**BMP**) image format. BMP is a simple way to store images in digital format. For every BMP file, there will be a header section, which will give a lot of information such as image resolution (number of pixels in horizontal and vertical directions), whether the image is colour or grayscale etc. to the application(software) which is used to view the image (such as MS paint, Photoshop etc.). After the header, there will be image data. You can think of it as a large matrix where each row represents each line in the image and each element in the row as a pixel. The number of lines in the image is basically the vertical resolution and number of pixels in a line is the horizontal resolution. For a colour image, each pixel (point) in the image is represented using 3 bytes (24 bits). 1 byte for Red, 1 for Green and 1 for Blue. Because of this, a colour image is usually called an RGB image. For a grayscale image, each pixel is represented using 1 byte. Grayscale images have no other colour, but different shades of black as shown in the figure below.

Fig.3 Grayscale Image

May be you usually call this as black and white image, but strictly speaking it is not since it has different shades of black (gray). A real black and white image is the one shown before as the output of the edge detection (Fig.2). It is also called a binary image. Since in grayscale image each pixel is represented using one byte (8 bits), each pixel can have 256 possible values (0 to 255). Here 0

represents black and 255 represents white, and all the values in between represent different shades of black.

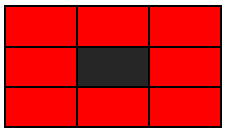
Image Processing

For edge detection, it is easier if the image is in grayscale rather than in RGB format. So your first task will be to convert the RGB image into grayscale image. How the 3 bytes representing R, G and B can be converted into a single byte representing gray level has been found after extensive research on the human eye (brain?) sensitivity to different colours. It is found that we are most sensitive to green, then red and finally blue. Based on this a formula is developed,

Y = 0.2126R + 0.7152G+0.0722B

where Y is the gray level (it is also called the brightness or luminance). Please note that when you convert RGB image to grayscale, the size of the new image will be roughly 1/3rd of the original.

Once you convert the image to grayscale, you will select one pixel from the image at a time and apply something called as *neighbourhood* operation. It means you will change the value of a pixel based on the values of its neighbours. For Sobel operation, generally we consider 8 neighbours.



In the figure, black represents the pixel we are processing and red represents its neighbours. Now we multiply the pixel and its neighbours with two matrices as shown below

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

where A represents the 3x3 matrix of the pixel and its neighbours. Please note that here you should not use the true matrix multiplication, but multiply only the corresponding elements (similar to matrix addition). You will calculate

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

The G value has to be calculated for every pixel in the grayscale image (so for a 512x512 resolution image, 262144 times!!). You will notice that if you sum the elements of G_x and G_y, it will be zero. That helps in finding the edge. More about how this is happening, please refer to online documents on Sobel filtering.

After finding the G values, you can compare it with a *threshold* value. If G is above the threshold, assign pixel as white (255), else black (0).

Prerequisites

You should understand binary file processing in Python for doing this project. You should clearly understand how data is stored in text files and binary files.

Guidelines

1. Initially understand the structure of BMP image files from online resources

2. To see how data is stored in image files, you can use the HxD-Hexeditor in Windows machine or Bless Hex editor in Linux. If you try to open image files in text editors (such as notepad++), you will be only seeing funny symbols.
3. Use the *struct* module in Python to convert binary data from the image file to required format (integers, string etc.)
4. You are strongly advised to use modular and object oriented programming styles while doing the project

Deliverables

You need to provide a *Python Module* called *sobel* as the project output. I should be able to import this module to my other Python code and use for edge detection. The module should provide the following functionalities.

1. A function to read a BMP image
2. A function to convert RGB image into grayscale
3. A function to apply Sobel edge detection to a grayscale image with a specific threshold
4. It will be great to have a function to display the image also (you can use matplotlib package)

I will be using your module like this

```
import sobel

I = sobel.imread('lena_color.bmp')   #imread() is the function to
read image.lena_color.bmp is the image
J = sobel.rgb2gray(I)  #rgb2gray is the function to convert RGB
image to grayscale
K = sobel.edgedetect(J,100)  #edgedetect() is the function for sobel
filtering and 100 is my threshold here
sobel.imshow(K)#imshow() is the function to display the image
```

Threshold 80

Threshold 150

Threshold 220