اسم الطالب: فهد بادي عبدالحميد جلال
رقم الجلوس: 6320
القسم (البرنامج): الإتصالات والإلكترونيات **الفرقة:** الثالثة
اسم المقرر: الرؤية الحاسوبية (مقرر اختياري) **كود المقرر:** هكت 331
اسم استاذ المقرر: دكتور مصطفي صلاح
اسم الملف الإلكترونى المرسل: PhotoSketching.pdf

عنوان الموضوع البحثي : Photo Sketching

الدرجة	ناجح	راسب	توقيع السادة أعضاء لجنة تصحيح المقرر	
			الاسم	التوقيع
----------	○	○	1- ----------	----------
100			2- ----------	----------
			3- ----------	----------

ختم شئون الطلاب

يعتمد رئيس القسم العلمي

جامعة سوهاج
كلية الهندسة
اسم الطالب: فهد بادي عبدالحميد جلال
رقم الجلوس: 6320

Photo Sketching

Abstract-

Open CV does not offer a native function to implement these techniques, but with a little insight and a few tricks, we will arrive at our own efficient implementation that can be used to produce a pencil sketch.

Keywords- Introduction to algorithm, Source code, Result.

INTRODUCTION TO STEPS AND ALGORITHMS

1. **Reading / Loading an image file:**
   Open CV provide the ***imread***( ) function that support various file formats for images. The supported formats vary by system but should always include the BMP format. Typical, PNG, JPEG, and TIFE should be among the supported formats too.
   By default ***imread***( ) returns an image in RGB color format, even if the file uses a gray-scale format.
   Note: BGR (blue-green-red) represents the same color space as **RGB** (red-green-blue) but the byte order is reversed.

2. **Convert the color image to gray-scale:**
   An RGB or BGR color image consists of 3 layers as it is clearly see through its name, it's a 3 dimensional matrix where gray-scale image is of only 2 dimensional, and the values ranges between 0-255 (8-bit unsigned integers).
   Therefore some algorithms can only applied on 2-D image, hence we covert an RGB image into a gray-scale image. For example, Convolution of an image, Black and white conversion of an image, etc.
   When converting an RGB image to gray-scale, we have to take the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One such approach is to lake the average of the contribution from each channel:

$$Gray = \frac{R + G + B}{3}$$

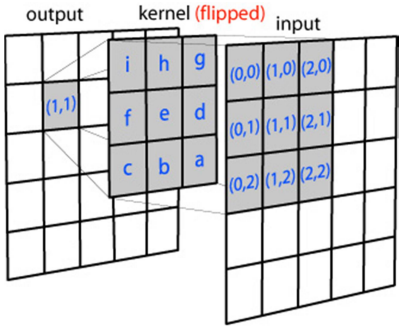
However, we do not see each color as a 'third' of the intensity but rather we see more green than red and blue. The final value is set to (approximately):

$$Gray = 0.3R + 0.59G + 0.11B$$

So that the single channel (gray-level) image best corresponds.
Open CV provide the ***cvtColor***( ) function, or by reading the image directly in gray format using function ***imread***(nameImage, 0).

3. **Invert the image to get a negative:**
   Know color inversion as the negative effect color inversion is achieved by subtracting each RGB color value from the maximum possible value (usually 255).
   A negative of the image can be obtained by 'inverting' the gray-scale value of every pixel. The invert of gray-scale value *x* is simple 255 - *x*.

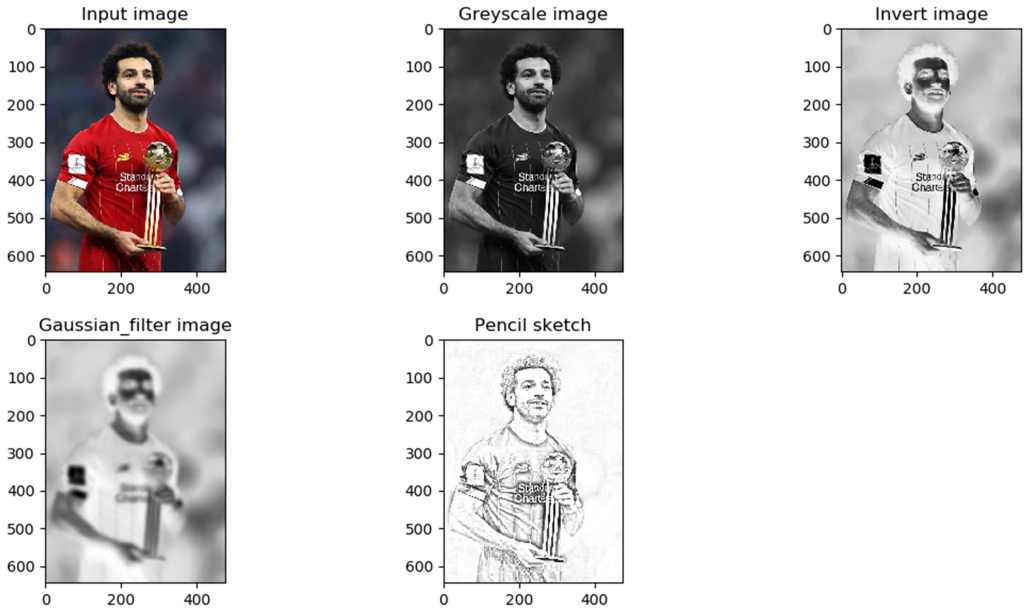
4. **Blur the image:**
   A Gaussian blur is an effective way to both reduce noise and reduce the amount of detail in an image. Mathematically it is equivalent to convolving an image with a Gaussian kernel.
   In convolution, two mathematical functions are combined to produce a third function. In image processing functions are usually called kernels. A kernel is nothing more than a square array of pixels. Usually the values in the kernel add up to one. This is to make sure no energy is added or removed from the image after the operation.
   Each pixel in the image gets multiplied by the Gaussian kernel. This is done by placing the center pixel of the kernel on the image pixel and multiplying the values in the original image with the pixels in the kernel that overlap. The values resulting from these multiplications are added up and that result is used for the value at the destination pixel. Looking at the image (Fig.1), you would multiply the value at (0, 0) in the input array by the value at (i) in the kernel array, the value at (1, 0) in the input array by the value at (h) in the kernel array, and so on, and then add all these values to get the value for (1, 1) at the output image.



Gaussian Blur "Fig.1"

5. **<u>Dodge blend the image</u>:**
   We use our dodging function to blend the gray-scale image with blurred invers image.
   Implementing dodging and burning:
   a) Determine the shape (width and height) of the input image.
   b) Prepare output argument with same size as input image.
   c) Shift image pixel value by 8 bits or multiplying the pixel value with the number 256.
   d) Divide by the inverse of the mask.
   e) Make sure resulting value > 255.

- **ENTIRE CODE IN ACTION**



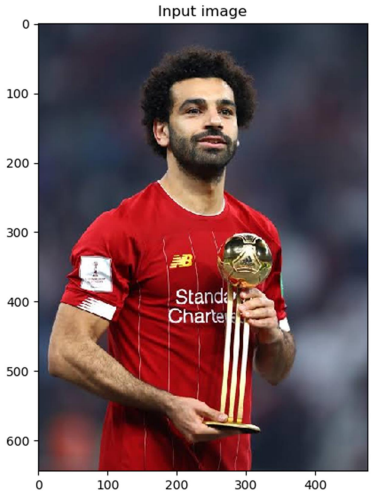
Each stage of the algorithm "Fig.2"

- **SOURCE CODE**

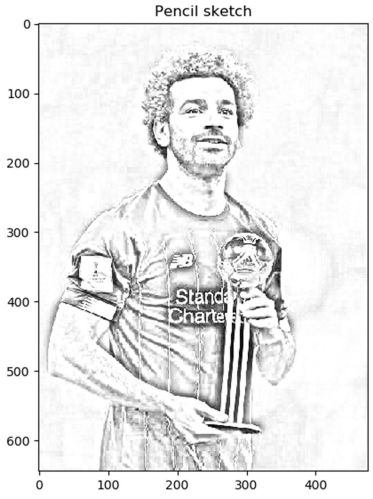
```python
from matplotlib import pyplot as plt
import numpy as np
import cv2

#Input image
input_img = cv2.imread('Mo_Salah.jpg')

#Convert the BGR to RGB because used plt.imshow() function.
b, g, r = cv2.split(input_img)
input_img = cv2.merge([r, g, b])

#convert the RGB color image to greyscale image
grey_img = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)

#Invert the greyscale image to get a negative
invert_img = 255-grey_img

#Gaussian blur to the negative from invert image
blur_img = cv2.GaussianBlur(invert_img, (21,21), 15)

#Blend the greyscale image with the Gaussian blur image using a color dodge
def dodgeColour(image, mask):
    return cv2.divide(image, 255-mask, scale=256)
pencil = dodgeColour(grey_img, blur_img)

#show input image and output image
plt.subplot(121), plt.imshow(input_img, cmap="gray"), plt.title('Input image')
plt.subplot(122), plt.imshow(pencil, cmap="gray"), plt.title('Pencil sketch')
plt.show()

```

Source code “Fig.3”

- **FINAL RESULT**



Input image “Fig.4-a”



Output image “Fig.4-b”

CONCLUSION

This paper discussed a student project to convert RGB image to pencil sketch image using open CV by python. The steps and algorithm was discussed along with the anticipated results.

REFERENCES

[1] OpenCV: Computer Vision Projects with Python / Joseph Howse, Prateek Joshi, Michael Beyeler - 2016.

[2] Simply OpenCV (Arabic) / Khaled Aldobosh – 2017.

[3] By Rishav Agarwal, “freeCodeCamp” Accessed on: July. 21, 2018. [online].

[4] By Michael Beyeler, “+Ask aSwiss” Accessed on: Janu. 13, 2016. [online].