GUIDED ASSIGNMENT ON EDGE DETECTION

Name: **Shreya Shrivastava**

Course: **Artificial Intelligence And Machine Learning**

Batch Four

Duration: 12 Months

Problem statement: Using OpenCV, first convert any image with varying High condition to a grayscale image. Now implement edge detection first using the canny edge detection. Then apply simple thresholding and also **Adaptive/OTSU thresholding using OpenCV** to see the working of each of these methods. Once you obtain good results, use the obtained edge detection result as a mask to give colour to all the edges (if edges use the colour from the original image, else leave it black only).

Prerequisites:

The libraries as well as things required in order for the program to work:

I. **Python 3.6** : The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url : https://www.anaconda.com/download/

II. **Additional Packages**: After you've downloaded anaconda or python, two more packages need to be installed i.e, numpy and matplotlib.If you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

- Numpy: pip install numpy
- Matplotlib: pip install matplotlib

If you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

- Numpy: conda install -c anaconda numpy
- Matplotlib: conda install -c anaconda matplotlib

III. **OpenCV :** OpenCV can be downloaded from the following url: https://sourceforge.net/projects/opencvlibrary/. It is strongly recommended to download OpenCV in a virtual environment.

METHODS USED:

A. OTSU THRESHOLDING

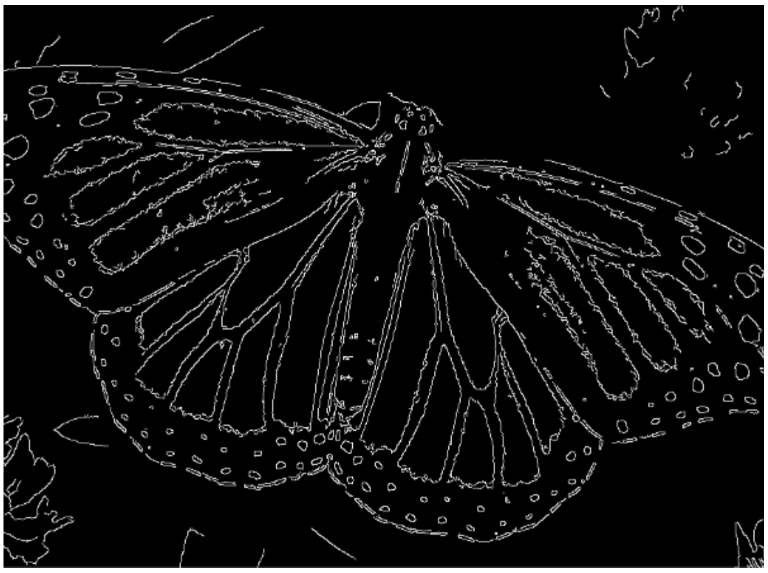
B. SIMPLE THRESHOLDING

C. CANNY EDGE DETECTION

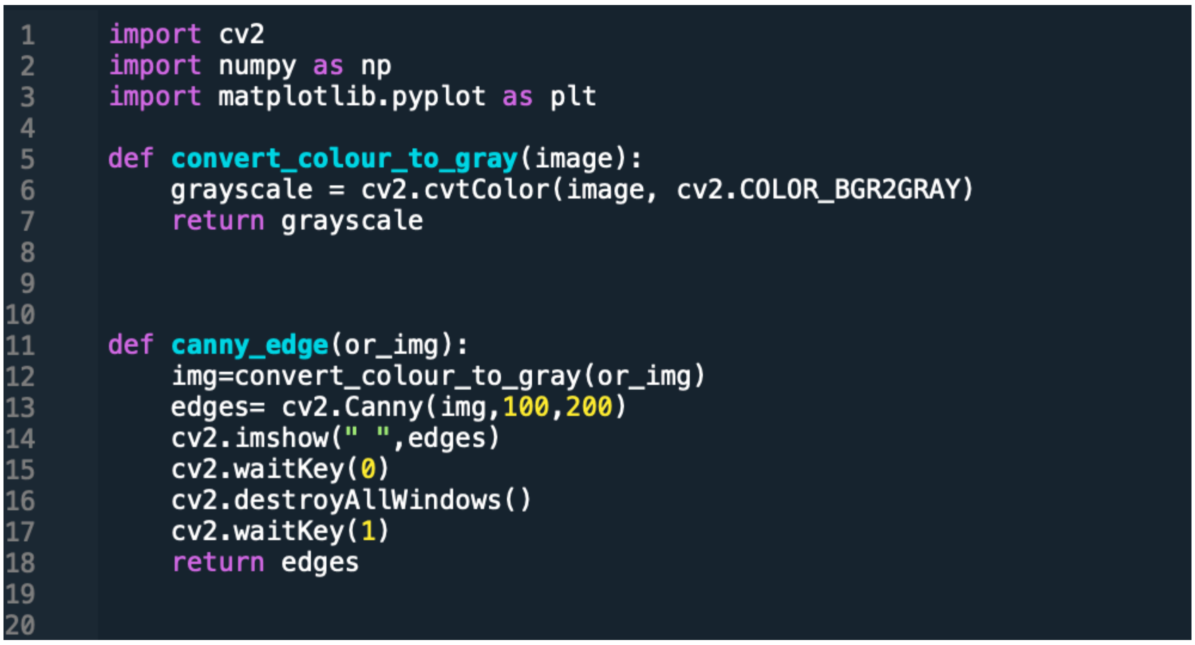
THE PROJECT :

1. Importing the libraries and performing ***Canny edge detection***. The ***convert_-colour_to_gray*** function takes the original coloured image as input and transforms it to a Grayscale image. The ***canny_edge*** function then calls the above function and performs edge detection as the name suggests displaying the now transformed image.

ORIGINAL IMAGE

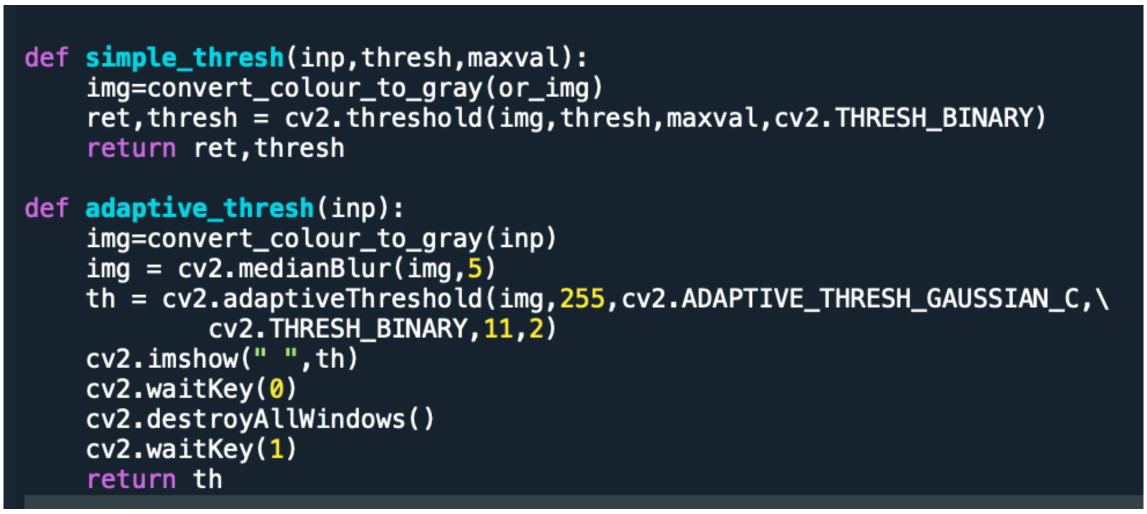


OUTPUT: CANNY DETECTION

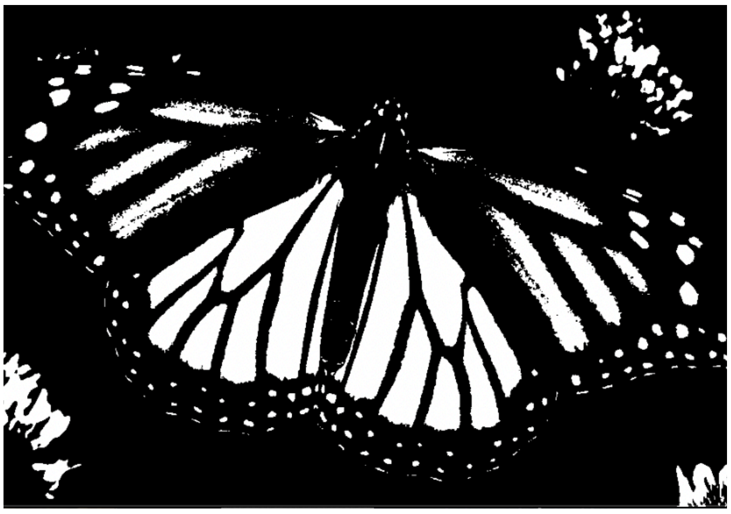


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def convert_colour_to_gray(image):
    grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return grayscale



def canny_edge(or_img):
    img=convert_colour_to_gray(or_img)
    edges= cv2.Canny(img,100,200)
    cv2.imshow(" ",edges)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    cv2.waitKey(1)
    return edges


```

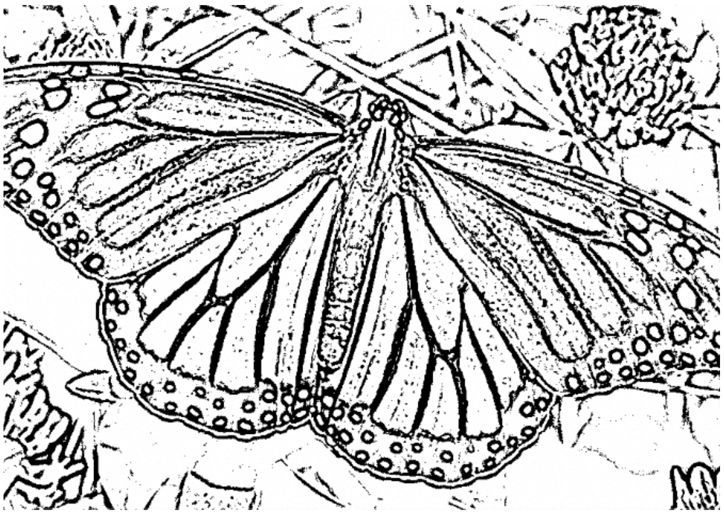
2. Here we perform ***simple*** as well as ***adaptive/OTSU thresholding***. Simple thresholding takes in the image as well the maximum value (which in our case is 255) and the threshold value(here as 140) as input.



```
def simple_thresh(inp,thresh,maxval):
    img=convert_colour_to_gray(or_img)
    ret,thresh = cv2.threshold(img,thresh,maxval,cv2.THRESH_BINARY)
    return ret,thresh

def adaptive_thresh(inp):
    img=convert_colour_to_gray(inp)
    img = cv2.medianBlur(img,5)
    th = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,2)
    cv2.imshow(" ",th)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    cv2.waitKey(1)
    return th
```

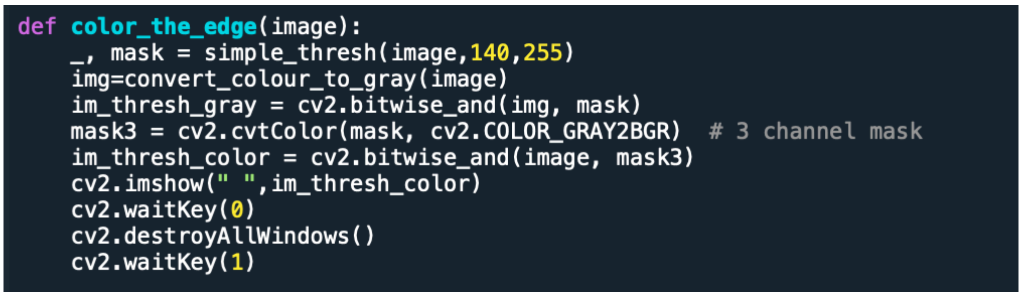


SIMPLE THRESHOLDING

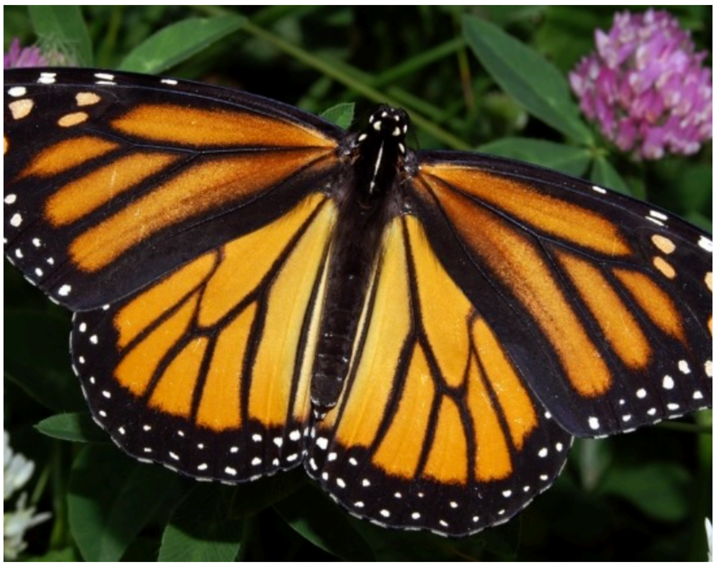


ADAPTIVE THRESHOLDING

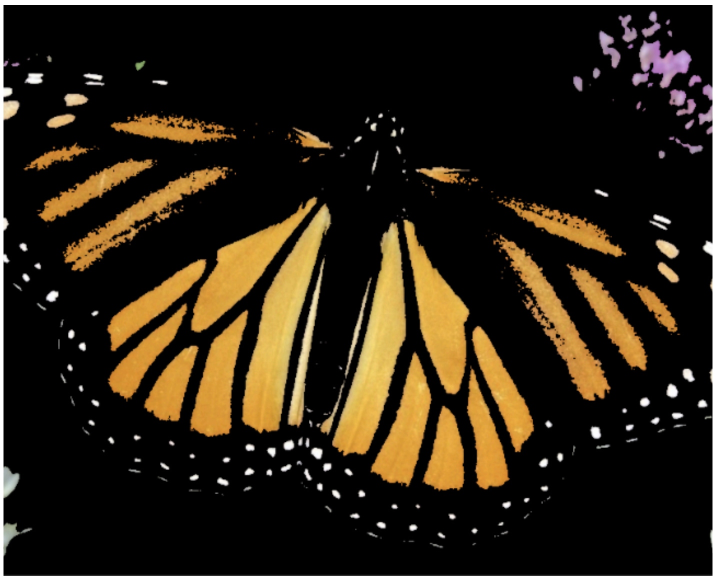
3. The final leg of the project required us to use the output of canny detection as the mask and colour the edges as in the original image. For this, we first performed a simple thresholding followed by the usage of the bitwise_and function.



```
def color_the_edge(image):
    _, mask = simple_thresh(image,140,255)
    img=convert_colour_to_gray(image)
    im_thresh_gray = cv2.bitwise_and(img, mask)
    mask3 = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)  # 3 channel mask
    im_thresh_color = cv2.bitwise_and(image, mask3)
    cv2.imshow(" ",im_thresh_color)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    cv2.waitKey(1)
```



ORIGINAL IMAGE



MASKING THE OUTPUT OF CANNY EDGE DETECTION ON THE ORIGINAL IMAGE AND RECOLOURING THE EDGES, SCRAPING OUT THE REST.