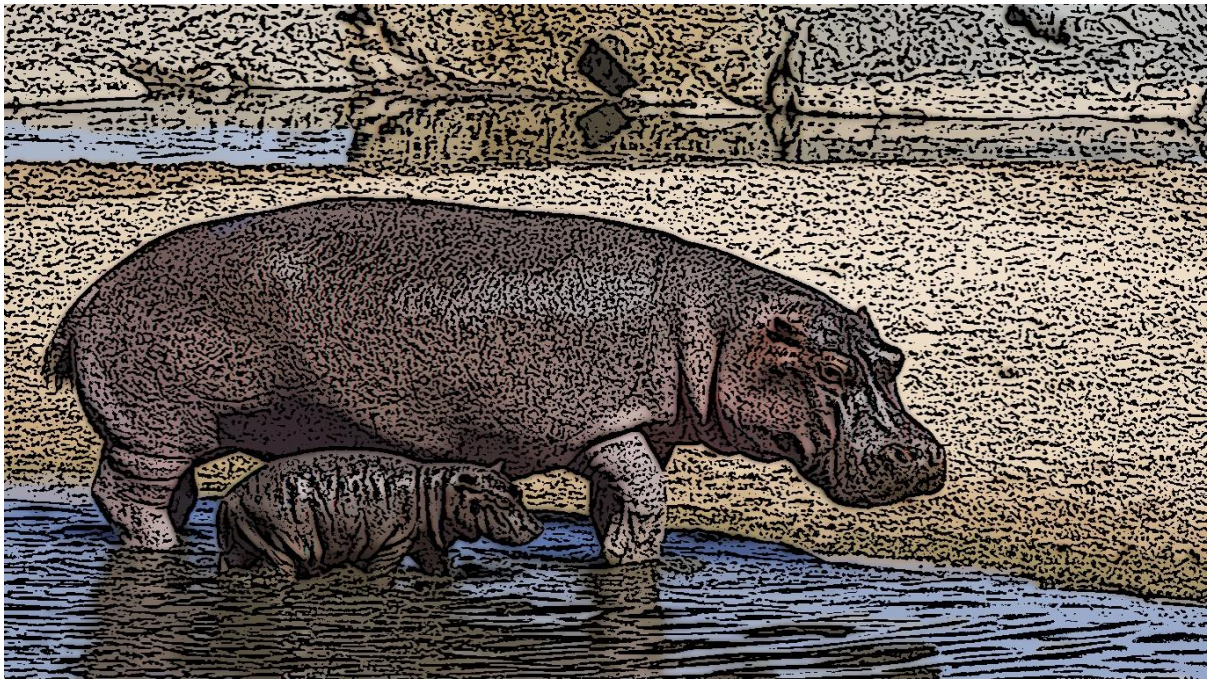
CodeBozu: Bozu Image Filter

Souradeep Pal, Group-46

Date: 11th March, 2022

After doing the project and working through the deliverables as an advanced student, I can now confidently work with the OpenCV and NumPy modules. I even got a basic understanding of matplotlib. I have learnt how to efficiently bring out or enhance different attributes of an image. I was very much interested by the concepts of kernels and filters and will continue to study on them. The project allowed me to work along specific guidelines, thus increasing my learning speed greatly. I would like to thank the CodeBozu mentors for this opportunity to learn and apply my code for practical uses.



Source Code:

Deliverable 1:

```
import cv2
import numpy as np

image=cv2.imread('Bozu.png')

def reddify(image):
    red_channel = image[:,:,2]
    red_img = np.zeros(image.shape)
    red_img[:,:,2] = red_channel
    cv2.imwrite('Red_Bozu.jpg',red_img)
    return red_img

def greenify(image):
    green_channel = image[:,:,1]
    green_img = np.zeros(image.shape)
    green_img[:,:,1] = green_channel
    cv2.imwrite('Green_Bozu.jpg',green_img)
    return green_img

def blueify(image):
    blue_channel = image[:,:,0]
    blue_img = np.zeros(image.shape)
    blue_img[:,:,0] = blue_channel
    cv2.imwrite('Blue_Bozu.jpg',blue_img)
    return blue_img

def grayify(image):
    h,w=image.shape[:2]
    gray_img = np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            (B,G,R)=image[i,j]
            #print('R={},G={},B={}'.format(R,G,B))
            gray_img[i,j] = 0.2989*R + 0.5870*G + 0.1140*B
    cv2.imwrite('Gray_Bozu.jpg',gray_img)
    return gray_img

def negative(image):
    negative_img = 255-image
    cv2.imwrite('Negative_Bozu.jpg',negative_img)
    return negative_img
```

Deliverable 2:

```
import numpy as np
import cv2

image=cv2.imread('Bozu.png')

def horizontal_flip(image):
    h,w=image.shape[:2]
    hflip_img=np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            hflip_img[i,j]=image[i,w-j-1]
    return hflip_img

def vertical_flip(image):
    h,w=image.shape[:2]
    vflip_img=np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            vflip_img[i,j]=image[h-i-1,j]
    return vflip_img

# cv2.imwrite('Horizontal_Red_Bozu.png',horizontal_flip(image))
# cv2.imwrite('Vertical_Red_Bozu.png',vertical_flip(image))


def clip(broken_image):
    h,w = broken_image.shape[:2]
    for i in range(h):
        for j in range(w):
            (B,G,R)=broken_image[i,j]
            broken_image[i,j,0] = 0 if B<0 else 255 if B>255 else B
            broken_image[i,j,1] = 0 if G<0 else 255 if G>255 else G
            broken_image[i,j,2] = 0 if R<0 else 255 if R>255 else R
    return broken_image

def contrast(image, alpha):
    h,w = image.shape[:2]
    contrast_img=np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            (B,G,R)=image[i,j]
            contrast_img[i,j,0] = B * alpha
            contrast_img[i,j,1] = G * alpha
            contrast_img[i,j,2] = R * alpha
            (B,G,R)=contrast_img[i,j]
            contrast_img[i,j,0] = 0 if B<0 else 255 if B>255 else B
            contrast_img[i,j,1] = 0 if G<0 else 255 if G>255 else G
```

```
            contrast_img[i,j,2] = 0 if R<0 else 255 if R>255 else R
            # contrast_img[i,j,0] = 0 if B<0 else 1 if B>1 else B
            # contrast_img[i,j,1] = 0 if G<0 else 1 if G>1 else G
            # contrast_img[i,j,2] = 0 if R<0 else 1 if R>1 else R
    cv2.imwrite('Contrast_Bozu.jpg',contrast_img)
    return contrast_img

#contrast(image,1.5)

def add_brightness(image, beta):
    h,w = image.shape[:2]
    bright_img=np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            #image[i,j]+=beta
            (B,G,R)=image[i,j]
            bright_img[i,j,0] = B + beta
            bright_img[i,j,1] = G + beta
            bright_img[i,j,2] = R + beta
            (B,G,R)=bright_img[i,j]
            bright_img[i,j,0] = 0 if B<0 else 255 if B>255 else B
            bright_img[i,j,1] = 0 if G<0 else 255 if G>255 else G
            bright_img[i,j,2] = 0 if R<0 else 255 if R>255 else R
    cv2.imwrite('Bright_Bozu.jpg',bright_img)
    return bright_img

#add_brightness(image,100)


def apply_threshold(image, threshold):
    threshold_img=np.zeros(image.shape)
    h,w=image.shape[:2]
    for i in range(h):
        for j in range(w):
            (B,G,R)=image[i,j]
            threshold_img[i,j,0] = 0 if B<threshold else 255
            threshold_img[i,j,1] = 0 if G<threshold else 255
            threshold_img[i,j,2] = 0 if R<threshold else 255
    cv2.imwrite('Bozu_in_the_dark.jpg',threshold_img)
    return threshold_img

#to get just the white eyes
apply_threshold(image,255)
```

Deliverable 5:

```python
import cv2
import numpy as np

#image=cv2.imread('Bozu.png')

def Blur(image, ksize):
    blurred_img = cv2.GaussianBlur(image, (ksize,ksize),0)
    cv2.imwrite('Blurred_Bozu.jpg', blurred_img)
    return blurred_img

#Blur(image,25)

def Vintage(image):
    h,w = image.shape[:2]
    k_y = cv2.getGaussianKernel(w,200)
    k_x = cv2.getGaussianKernel(h,200)
    kernel = k_y * k_x.T
    filter = 255 * kernel / np.linalg.norm(kernel)
    vintage_img = np.zeros(image.shape)
    for i in range(h):
        for j in range(w):
            vintage_img[i,j,0] = image[i,j,0] * filter[j][i]
            vintage_img[i,j,1] = image[i,j,1] * filter[j][i]
            vintage_img[i,j,2] = image[i,j,2] * filter[j][i]
    cv2.imwrite('Vintage_Bozu.jpg',vintage_img)
    return vintage_img

#Vintage(image)

def Sepia(image):
    sepia_img=np.zeros(image.shape)
    h,w = image.shape[:2]
    for i in range(h):
        for j in range(w):
            B,G,R=image[i,j]
            sepia_img[i,j,0] = R*0.272 + G*0.534 + B*0.131
            sepia_img[i,j,1] = R*0.349 + G*0.686 + B*0.168
            sepia_img[i,j,2] = R*0.393 + G*0.769 + B*0.189
            (B,G,R)=sepia_img[i,j]
            sepia_img[i,j,0] = 0 if B<0 else 255 if B>255 else B
            sepia_img[i,j,1] = 0 if G<0 else 255 if G>255 else G
            sepia_img[i,j,2] = 0 if R<0 else 255 if R>255 else R
    cv2.imwrite('Sepia_Bozu.jpg',sepia_img)
    return sepia_img

#Sepia(image)
```

```
def Sharpen(image):
    kernel=np.array([[-1, -1, -1], [-1, 9.5, -1], [-1, -1, -1]])
    sharp_img = cv2.filter2D(image, -1, kernel)
    cv2.imwrite('Sharp_Bozu.jpg',sharp_img)
    return sharp_img

#Sharpen(Sepia(image))
```

Deliverable 6:

```
import cv2
from matplotlib.pyplot import gray
import numpy as np
from Deliverable_1 import *
from Deliverable_2 import *
from Deliverable_5 import *

def render(image):
    gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blur_img = Blur(gray_img, 3)

    edge_img = cv2.adaptiveThreshold(blur_img, 255,
                                     cv2.ADAPTIVE_THRESH_MEAN_C,
                                     cv2.THRESH_BINARY, 9, 2)

    (x, y)= image.shape[:2]
    edge_img = cv2.resize(edge_img, (y, x))
    edge_img = cv2.cvtColor(edge_img, cv2.COLOR_GRAY2RGB)
    cv2.imwrite("edge.png", edge_img)
    return cv2.bitwise_and(image, edge_img)

image=cv2.imread('MyPic.jpeg')
res = render(image)
```

THANK YOU