**بسمه تعالی**

**عنوان تمرین:**

**پردازش تصویر**

**نام دانشجو:**

**..**

**شماره دانشجویی:**

**....**

**استاد:**

**...**

پیاده سازی در پایتون

پیاده سازی

ابتدا کتابخانه های مورد نیاز را بارگذاری می کنیم

from **PIL** import **Image**

import **numpy** as **np**

from **skimage**.**metrics** import **structural\_similarity** as **ssim**

import **cv2**

import **matplotlib**.**pyplot** as **plt**

from **skimage**.**feature** import **hessian\_matrix**, **hessian\_matrix\_eigvals**

import **random**

import **pandas** as **pd**

**سوال 1 :**

ابتدا باید چرخش را حذف نمود



برای این منظور ابتدا عکس را میچرخانیم تا زاویه مناسب بدست آید

سپس با زاویه مناسب چرخش داده و قسمت های اضافه عکس را برش می زنیم



path\_one="Ex1\_Data/One/"

*# creating a image object (main image)*

finger2 = **Image**.**open**(r"Ex1\_Data\One\2.bmp").**convert**("L")

*# rotating a image 90 deg counter clockwise*

finger2 = finger2.**rotate**(-63,**Image**.NEAREST, expand = 1)

finger2 = finger2.**crop**((43, 39,139 , 142))

finger2.**save**(path\_one+"2.bmp")

finger2.**show**()

*# creating a image object (main image)*

finger1 = **Image**.**open**(r"Ex1\_Data\One\1.bmp").**convert**("L")

*# rotating a image 90 deg counter clockwise*

finger1 = finger1.**rotate**(-45,**Image**.NEAREST, expand = 1)

finger1 = finger1.**crop**((53,51,149 , 154))

finger1.**save**(path\_one+"1.bmp")

finger1.**show**()

**برای مقایسه عکس دو ویژگی PSNR و SSIM را برای هر اثر انگشت بکار میبریم**

**SSIM :**

similar\_finger1=[]

for n in **range**(1,21):

    path='Ex1\_Data/TwentyFingerprints/'+**str**(n)+".bmp"

    other\_fin=**Image**.**open**(path)

    other\_fin=other\_fin.**convert**("L")

    other\_fin=**np**.**array**(other\_fin)

    similar\_finger1.**append**(**ssim**(**np**.**array**(finger1),other\_fin))

**print**("finger ",similar\_finger1.**index**(**max**(similar\_finger1))+1,**max**(similar\_finger1))

similar\_finger1

**PSNR :**

def **PSNR**(original, n):

    path='Ex1\_Data/TwentyFingerprints/'+**str**(n)+".bmp"

    other\_fin=**Image**.**open**(path).**convert**("L")

    other\_fin=**np**.**array**(other\_fin)

    mse = **np**.**mean**((original - other\_fin) \*\* 2)

    if(mse == 0(:

        return 100

    max\_pixel = 255.0

    psnr = 20 \* **np**.log10(max\_pixel / **np**.sqrt(mse))

    return psnr

با استفاده از کد های بالا هر کدام هر کدام از اثر انگشت ها را برای 20 اثر انگشت مقایسه میکنیم و درون یک آرایه خالی میریزیم

سپس اندیس بیشتر مقدار هر آرایه به اضافه یک برابر شماره اثر انگشت خواهد بود

**سوال 2 :**

ابتدا با استفاده از تابعی که خود مسئله در اختیار ما گذاشته ، عکس های خاکستری را رنگی می کنیم

def **colorizeBy\_colorPalette**(image\_addr:**str**, colorPalette\_addr:**str**, output\_addr:**str** = "."):

    grayimg = **Image**.**open**(image\_addr)

    grayimg\_arr = **np**.**array**(grayimg)

    colors\_data = **pd**.**read\_csv**(colorPalette\_addr)

    R = **np**.**zeros**(grayimg\_arr.shape)

    G = **np**.**zeros**(grayimg\_arr.shape)

    B = **np**.**zeros**(grayimg\_arr.shape)

    for col in **range**(grayimg\_arr.shape[1]):

        for row in **range**(grayimg\_arr.shape[0]):

            R[row, col] = colors\_data.iloc[grayimg\_arr[row, col], 0]

            G[row, col] = colors\_data.iloc[grayimg\_arr[row, col], 1]

            B[row, col] = colors\_data.iloc[grayimg\_arr[row, col], 2]

    out = **np**.**array**((R, G, B)).transpose(1, 2, 0)

    new = **Image**.**fromarray**(out.astype('uint8'), 'RGB')

    new.**save**(f'{output\_addr}/Image\_color.jpg')

    return new

path="D:/dev/python/penhan ax/Ex1\_Data/Two\_A"

img1=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_A/Forest\_1/Forest\_1\_S.jpg")

img2=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_A/Forest\_2/Forest\_2\_S.jpg")

img3=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_A/Forest\_3/Forest\_3\_S.jpg")

img1\_colorize=**colorizeBy\_colorPalette**(path+"/Forest\_1/Forest\_1\_S\_gray.jpg",path+"/Forest\_1/Forest\_1\_S\_color\_palette\_sort.csv",path+"/Forest\_1")

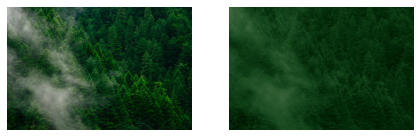
img2\_colorize=**colorizeBy\_colorPalette**(path+"/Forest\_2/Forest\_2\_S\_gray.jpg",path+"/Forest\_2/Forest\_2\_S\_color\_palette\_sort.csv",path+"/Forest\_2")

img3\_colorize=**colorizeBy\_colorPalette**(path+"/Forest\_3/Forest\_3\_S\_gray.jpg",path+"/Forest\_3/Forest\_3\_S\_color\_palette\_sort.csv",path+"/Forest\_3")

**plot\_images**(img1,img1\_colorize)

**plot\_images**(img2,img2\_colorize)

**plot\_images**(img3,img3\_colorize)

****

****

مقایسه تصویر اصلی با تصویر رنگ شده با پلت رنگی

سپس برای ساخت پلت رنگی از کد زیر استفاده می کنیم

def **palette**(img):

    arr = **np**.asarray(img)

    palette, index = **np**.**unique**(**asvoid**(arr).ravel(), return\_inverse=True)

    palette = palette.view(arr.dtype).reshape(-1, arr.shape[-1])

    count = **np**.**bincount**(index)

    order = **np**.**argsort**(count)

    return palette[order[::-1]]

def **asvoid**(arr):

    arr = **np**.ascontiguousarray(arr)

    return arr.view(**np**.**dtype**((**np**.void, arr.dtype.itemsize \* arr.shape[-1])))

در این کد ابتدا عکس را به آرایه نامپای تبدیل می کنیم و سپس براساس تعداد تکرار رنگ ها یا همان نمودار هیستوگرام رنگ ها را مرتب کرده و در یک آرایه بر میگرداند.

img = **Image**.**open**("Ex1\_Data\Two\_B\Forest\_1\_S.jpg", 'r').**convert**('RGB')

**print**(**palette**(img))

img\_platted=**palette**(img)[:256]

image=**Image**.**fromarray**(img\_platted).**save**("Ex1\_Data\Two\_B\Forest\_1\_palette.png")

df = **pd**.**DataFrame**(img\_platted, columns = ['R','G','B'])

df.**to\_csv**("Ex1\_Data\Two\_B\Forest\_1\_S\_color\_palette\_sort.csv",index=False)

ابتدا عکس را به آرایه تبدیل کردیم

خروجی تابع palette را یک آرایه است که تنها به 256 رنگ اصلی و پرتکرار آن نیاز داریم

سپس آن آرایه را به عکس و هم با استفاده کتابخانه پانداس با فایل اکسل تبدیل می کنیم

سپس دوباره با کد زیر عکس ها را رنگ آمیزی می کنیم

path="D:/dev/python/penhan ax/Ex1\_Data/Two\_B/"

img1=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_B/Forest\_1\_S.jpg")

img2=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_B/Forest\_2\_S.jpg")

img3=**Image**.**open**(r"D:/dev/python/penhan ax/Ex1\_Data/Two\_B/Forest\_3\_S.jpg")

img1\_colorize=**colorizeBy\_colorPalette**(path+"Forest\_1\_S.jpg",path+"Forest\_1\_S\_color\_palette\_sort.csv",path+"/Forest\_1")

img2\_colorize=**colorizeBy\_colorPalette**(path+"Forest\_2\_S.jpg",path+"Forest\_2\_S\_color\_palette\_sort.csv",path+"/Forest\_2")

img3\_colorize=**colorizeBy\_colorPalette**(path+"Forest\_3\_S.jpg",path+"Forest\_3\_S\_color\_palette\_sort.csv",path+"/Forest\_3")

**plot\_images**(img1,img1\_colorize)

**plot\_images**(img2,img2\_colorize)

**plot\_images**(img3,img3\_colorize)

**سوال 3:**

ابتدا از اثر انگشت خود عکس تهیه می کنیم و آن با استفاده از کد زیر نشان می دهیم

myfinger=**cv2**.**imread**("Ex1\_Data/my finger/myfinger2.png")

image = **cv2**.**cvtColor**(myfinger, **cv2**.COLOR\_BGR2RGB)

**plt**.**imshow**(image)

**plt**.**axis**("off")



سپس باید اثر انگشت را از آن استخراج کنیم

def **main**():

*# -------------------------- Step 1: import the image whose background has been removed ----------*

    img = **cv2**.**imread**("Ex1\_Data/my finger/myfinger2.png",1)

*# -------------------------- Step 2: Sharpen the image -------------------------------------------*

    kernel = **np**.**array**([[-1,-1,-1],

                   [-1, 9,-1],

                   [-1,-1,-1]])

    sharpened = **cv2**.**filter2D**(img, -1, kernel)

*# cv2.imshow("sharpened",sharpened)*

*# --------------------------- Step 3: Grayscale the image------------------------------------------*

    gray = **cv2**.**cvtColor**(sharpened,**cv2**.COLOR\_BGR2GRAY)

*# cv2.imshow("gray",gray)*

*# --------------------------- Step 4: Perform histogram equilisation ------------------------------*

    hist,bins = **np**.**histogram**(gray.flatten(),256,[0,256])

    cdf = hist.cumsum()

    cdf\_normalized = cdf \* hist.max()/ cdf.max()

    cdf\_m = **np**.**ma**.**masked\_equal**(cdf,0)

    cdf\_m = (cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())

    cdf = **np**.**ma**.**filled**(cdf\_m,0).**astype**('uint8')

    img2=cdf[gray]

*# ----------------------------- Step 5: Ridge detection filter ------------------------------------*

*#sigma = 2.7*

    a, b = **detect\_ridges**(img2, sigma=2.7)

**plot\_images**(a, b)

*# ----------------------------- Step 6: Convert image to binary image -----------------------------*

    img = **cv2**.**imread**("Ex1\_Data/my finger/fig1.jpg",0)

*# cv2.imshow("img",img)*

    bg = **cv2**.**dilate**(img,**np**.**ones**((5,5),dtype=**np**.uint8))

    bg = **cv2**.**GaussianBlur**(bg,(5,5),1)

*# cv2.imshow("bg",bg)*

    src\_no\_bg = 255 - **cv2**.**absdiff**(img,bg)

*# cv2.imshow("src\_no\_bg",src\_no\_bg)*

    ret,thresh = **cv2**.**threshold**(src\_no\_bg,240,255,**cv2**.THRESH\_BINARY)

*#cv2.imshow("threshold",thresh)*

**plot\_images**(thresh, thresh)

**main**()

* ابتدا عکس را بارگذاری می کنیم
* سچس یک ماتریس 3\*3 را در تصویر ضرب می کنید ( کانولوشن 2 بعدی بر روی تصویر ایجاد می کنیم)
* منفی بودن اعداد داخل ماتریس باعث میشود که تغییرات در مرز های تصویر بیشتر قابل مشاهده شود
* سپس نرمال سازی هیستوگرام انجام می دهیم
* یک فیلتر گوسی روی آن اعمال می کنیم



سپس با استفاده از کد زیر اثر انگشت را درون عکس مخفی می کنیم

*# Python program to demonstrate*

*# image steganography using OpenCV*

import **cv2**

import **numpy** as **np**

import **random**

*# Encryption function*

def **encrypt**():

*# img1 and img2 are the*

*# two input images*

    img2 = **cv2**.**imread**('Ex1\_Data/my finger/fig1.jpg')

*# img2 = cv2.resize(img2,(640,640), interpolation = cv2.INTER\_AREA)*

    img1 = **cv2**.**imread**('Ex1\_Data/Three\_A/Face\_gray.jpg')

    for i in **range**(img2.shape[0]):

        for j in **range**(img2.shape[1]):

            for l in **range**(3):

*# v1 and v2 are 8-bit pixel values*

*# of img1 and img2 respectively*

                v1 = **format**(img1[i][j][l], '08b')

                v2 = **format**(img2[i][j][l], '08b')

*# Taking 4 MSBs of each image*

                v3 = v1[:4] + v2[:4]

                img1[i][j][l]= **int**(v3, 2)

**cv2**.**imwrite**('pic3in2.png', img1)

*# Decryption function*

def **decrypt**(path='pic3in2.png'):

*# Encrypted image*

    img = **cv2**.**imread**(path)

    width = img.shape[0]

    height = img.shape[1]

*# img1 and img2 are two blank images*

    img1 = **np**.**zeros**((width, height, 3), **np**.uint8)

    img2 = **np**.**zeros**((width, height, 3), **np**.uint8)

    for i in **range**(width):

        for j in **range**(height):

            for l in **range**(3):

                v1 = **format**(img[i][j][l], '08b')

                v2 = v1[:4] + **chr**(**random**.randint(0, 1)+48) \* 4

                v3 = v1[4:] + **chr**(**random**.randint(0, 1)+48) \* 4

*# Appending data to img1 and img2*

                img1[i][j][l]= **int**(v2, 2)

                img2[i][j][l]= **int**(v3, 2)

*# These are two images produced from*

*# the encrypted image*

**cv2**.**imwrite**('pic2\_re.png', img1)

**cv2**.**imwrite**('pic3\_re.png', img2)

*# Driver's code*

**encrypt**()

**decrypt**()

خروجی بصورت روبرو خواهد شد :



و عکس های استخراج شده از عکس ترکیب شده بصورت زیر می باشند

همانطور که مشاهده می کنید عکس ها دارای کمی نویز می باشند.

برای افزایش و یا کاهش

استفاده از روش بیت پلن و ارزش گذاری هر کدام از بیت ها

def **bitplanes**(im):

    im = **Image**.**open**(im).**convert**("L")

    data = **np**.**array**(im)

    out = []

    for k in **range**(7,-1,-1):

        res = data // 2\*\*k & 1

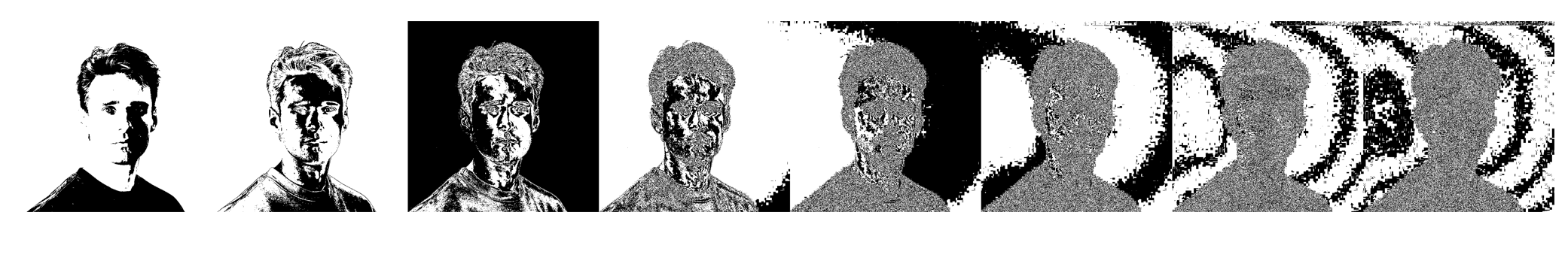
        out.**append**(res\*255)

*# stack generated images*

    b = **np**.**hstack**(out)

    return **Image**.**fromarray**(b)

**bitplanes**("Ex1\_Data/Three\_B/Comb\_Finger\_line.jpg").**show**()



secret = **Image**.**open**("Ex1\_Data/Three\_B/Comb\_Finger\_line.jpg")

data\_s = **np**.**array**(secret)

bit = **np**.bitwise\_and(**np**.right\_shift(data\_s, 1), 1)

data\_s = bit ^ data\_s & 1

new\_img = **Image**.**fromarray**(data\_s \* **np**.uint(255))

new\_img.**show**(

