دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

نام و نام خانوادگی:

پیمان هاشمی

شماره دانشجویی:

400131032

شماره تمرین:

تمرین شماره 4

درس:

تصویر پردازی رقمی

با سلام و عرض ادب

جناب آقای عباسی بابت تاخیر در ارسال گزارش عذرخواهی میکنم. متاسفانه یکی از اقوام نزدیک فوت کردن و مجبور شدم همراه با خانواده برای چندین روز به شهرستان بروم.

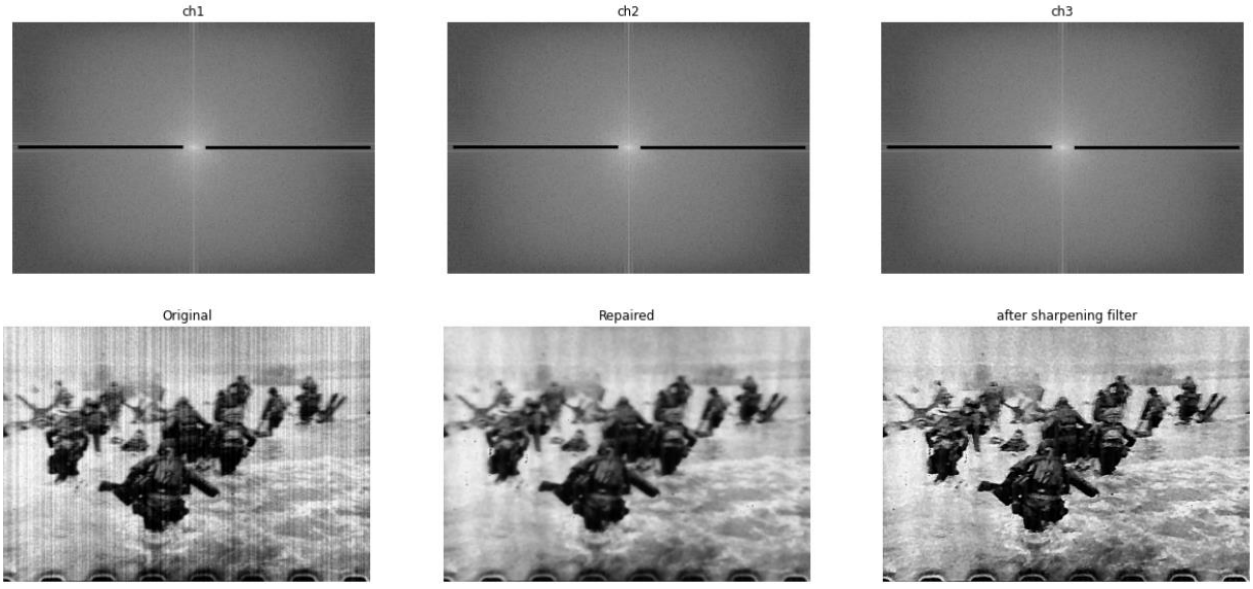
خواهشمندم در صورت امکان مساعدت بفرمایید و مدت تاخیر را نادیده بگیرید.

باتشکر

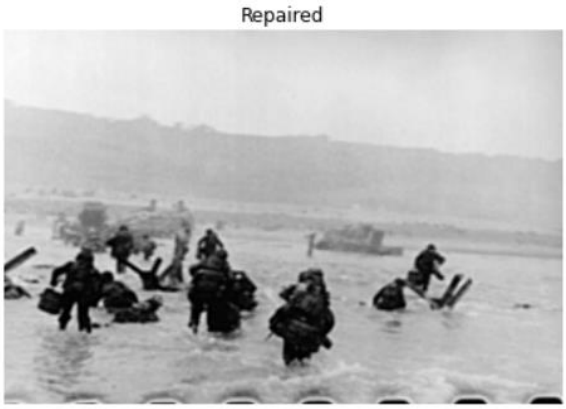
سوال 1)

A) در ابتدا فوریه تصویر را حساب کرده و با توجه به اینکه در تصویر نویز های عمودی وجود دارد، نویز های افقی در تبدیل فوریه را حذف میکنیم و سپس عکس را sharp میکنیم و از کرنل زیر برای شارپ کردن تصویر استفاده میکنیم.

-1	-2	-1
-2	13	-2
-1	-2	-1



B)در این قسمت نیز از sharp کردن تصویر استفاده میکنیم وکرنل مورد استفاده مانند بالا است.



C)

یک تابع به اسم final_denoiser تعریف کرده که در ابتدا عکس را با denoise_tv_chambolle نرمالایز کرده و سپس با استافده از فیلتر fastNlMeansDenoising نویز های باقی را از بین میبریم و سپس در یک فیلتر bilateral بر روی عکس اجرا میکنیم.

D)

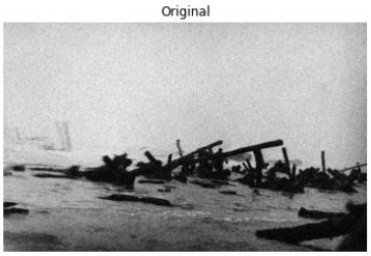
در ابتدا داده ها نرمالایز کرده و سپس از یک فیلتر دو بعدی که کرنل آن در زیر است عبور میدهیم.سپس از طریق fastNlMeansDenoising عکس را دنویز میکنیم.

E)

از تابع final_denoiser و عمل شارپ کردن استفاده میکنیم

F)

از تابع final_denoiser و عمل شارپ کردن استفاده میکنیم



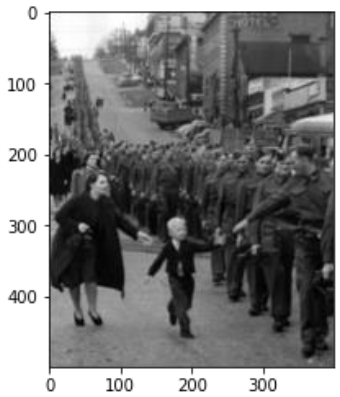
سوال 2)

A) در ابتدا اندازه تصویر را دو برابر میکنیم:


```
PSNR value is 30.909728869530287 dB
```

(B

```python
def bilinear_interpolation(image, y, x):
    height = image.shape[0]
    width = image.shape[1]

    x1 = max(min(math.floor(x), width - 1), 0)
    y1 = max(min(math.floor(y), height - 1), 0)
    x2 = max(min(math.ceil(x), width - 1), 0)
    y2 = max(min(math.ceil(y), height - 1), 0)

    a = float(image[y1, x1])
    b = float(image[y2, x1])
    c = float(image[y1, x2])
    d = float(image[y2, x2])

    dx = x - x1
    dy = y - y1

    new_pixel = a * (1 - dx) * (1 - dy)
    new_pixel += b * dy * (1 - dx)
    new_pixel += c * dx * (1 - dy)
    new_pixel += d * dx * dy
    return round(new_pixel)
```



PSNR value is 30.78672012597252 dB

(C

```
def euclidian_dist(a,b):
    return np.sqrt(((a[0]-b[0])**2)+((a[1]-b[1])**2))
```

```
def NN(X,P):

    i,j = X[0],X[1]
    A = [[i,j],[i,j+1],[i+1,j],[i+1,j+1]]
    dist = [euclidian_dist(A[0],P),euclidian_dist(A[1],P),euclidian_dist(A[2],P),euclidian_dist(A[3],P)]
    minpos = dist.index(min(dist))
    return A[minpos]
```

```
def NN_value_interpolation(im,scale_factor):

    row, col = im.shape[0], im.shape[1]
    n_row,n_col = int(scale_factor * row),int(scale_factor * col)
    zoom = np.arange(n_row*n_col).reshape(n_row,n_col)
    print("zoom shape is: ",zoom.shape,"image shape is: ", im.shape,'\n')
    for i in range(n_row):
        for j in range(n_col):
            P = [floor(float(i)/scale_factor),floor(float(j)/scale_factor)]
            X = [int(i) for i in P]
            zoom[i][j] = im[NN(X,P)[0]][NN(X,P)[1]]
    return zoom
```

PSNR value is 25.027313181900475 dB

(D

```
def non_uniform(arr):
    h, w, channel = arr[0].shape
    nh, nw = h * 2, w * 2
    blank = np.zeros((nh, nw, channel), dtype=np.uint8)
    dists = []

    dists.append([-0.3, 0.4])
    dists.append([0.6, 0.2])
    dists.append([-0.2, 0.3])
    dists.append([0.1, -0.5])

    dists_norm = [np.linalg.norm(ele) for ele in dists]

    summation = sum(dists_norm)
    dist_from_all = np.subtract([summation] * 4, dists_norm)
    temp = sum(dist_from_all)

    for k in range(channel):
        for i in range(nh):
            for j in range(nw):
                blank[i, j, k] = (arr[0][i // 2, j // 2, k] * dist_from_all[0] +
                                  arr[1][i // 2, j // 2, k] * dist_from_all[1] +
                                  arr[2][i // 2, j // 2, k] * dist_from_all[2] +
                                  arr[3][i // 2, j // 2, k] * dist_from_all[3]) / temp

    return blank
```

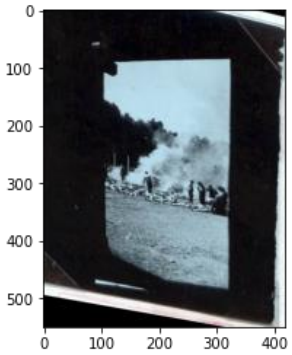
PSNR value is 22.783313181900475 dB

سوال 3)

A)

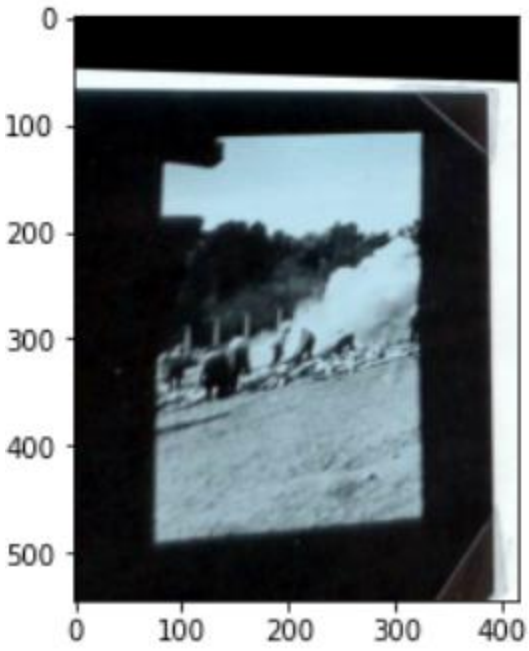
تصویر 1

```
# to calculate the transformation matrix
input_img = np.float32([[130,131],[317,85],[121,515],[320,473]])
output_img = np.float32([[98,81],[320,82],[95,477],[318,477]])

# Compute the perspective transform M
transform_M = cv2.getPerspectiveTransform(input_img,output_img)

# Apply the perspective transformation to the image
res = cv2.warpPerspective(img,transform_M,(img.shape[1], img.shape[0]),flags=cv2.INTER_LINEAR)

# Display the transformed image
plt.imshow(res)
```



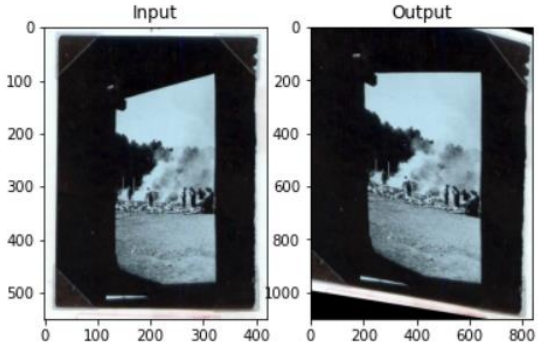
تصویر 2

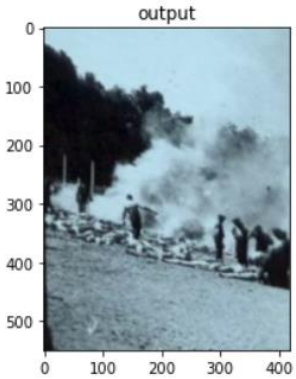
```
# to calculate the transformation matrix
input_img = np.float32([[169,37],[340,39],[190,378],[322,440]])
output_img = np.float32([[110,91],[340,92],[109,485],[310,485]])

# Compute the perspective transform M
transform_M = cv2.getPerspectiveTransform(input_img,output_img)

# Apply the perspective transformation to the image
res = cv2.warpPerspective(img,transform_M,(img.shape[1], img.shape[0]),flags=cv2.INTER_LINEAR)

# Display the transformed image
plt.imshow(res)
```


0
100
200
300
400
500
0
100
200
300
400

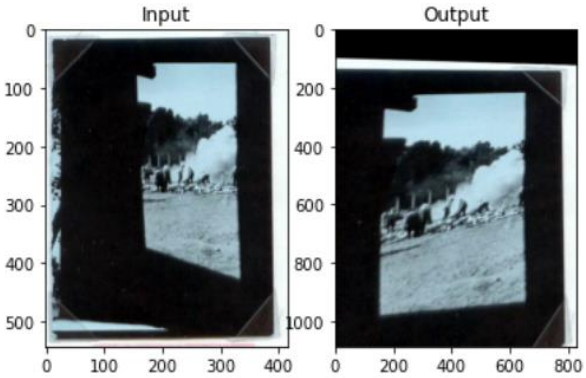
B)

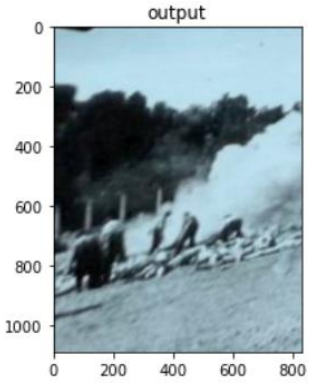
تصوير 1


Input
Output
0
100
200
300
400
500
0
100
200
300
400
0
200
400
600
800
1000
0
200
400
600
800


output
0
100
200
300
400
500
0
100
200
300
400

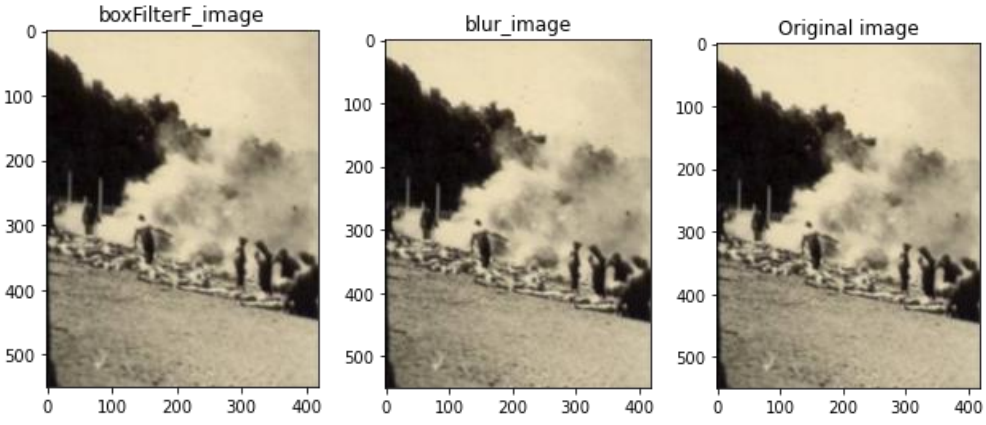
تصوير 2

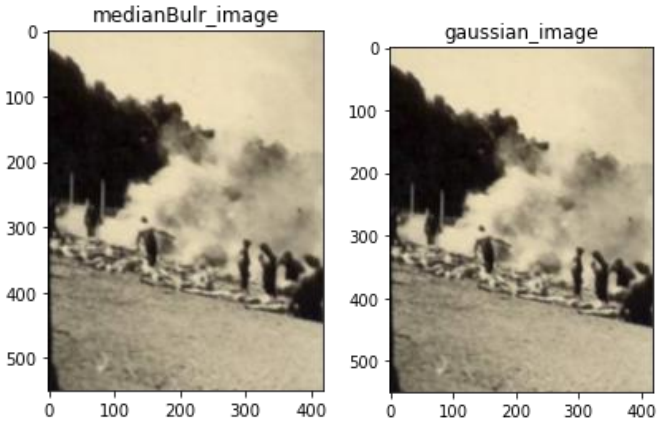




C)

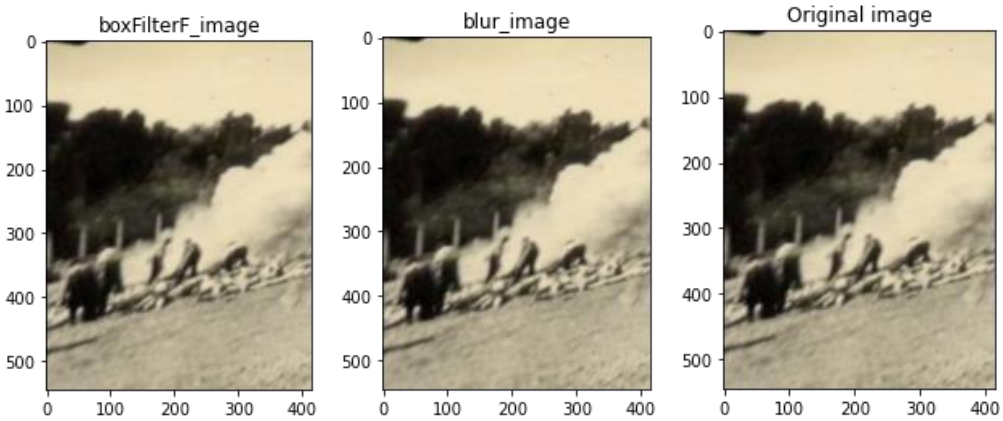
تصوير 1

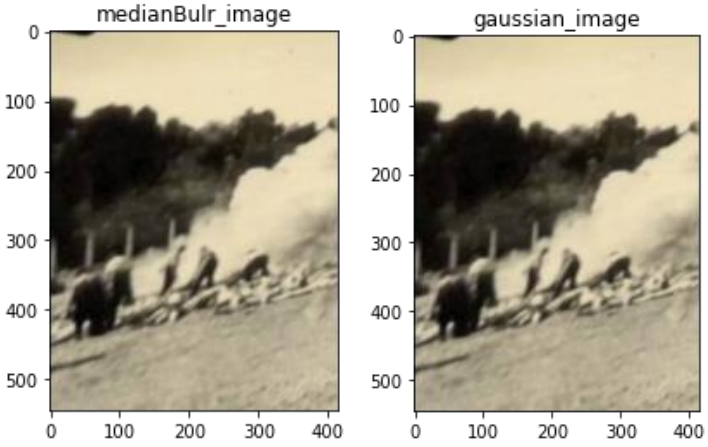



medianBulr_image
gaussian_image

x

تصوير 2


boxFilterF_image
blur_image
Original image


medianBulr_image
gaussian_image

بهتر کردن کیفیت

برای این کار ابتدا با کرنلی که در کد آمده تصویر را شارپ کرده و سپس با استفاده از fastNlMeansDenoising دوباره تصویر را دینویز میکنیم

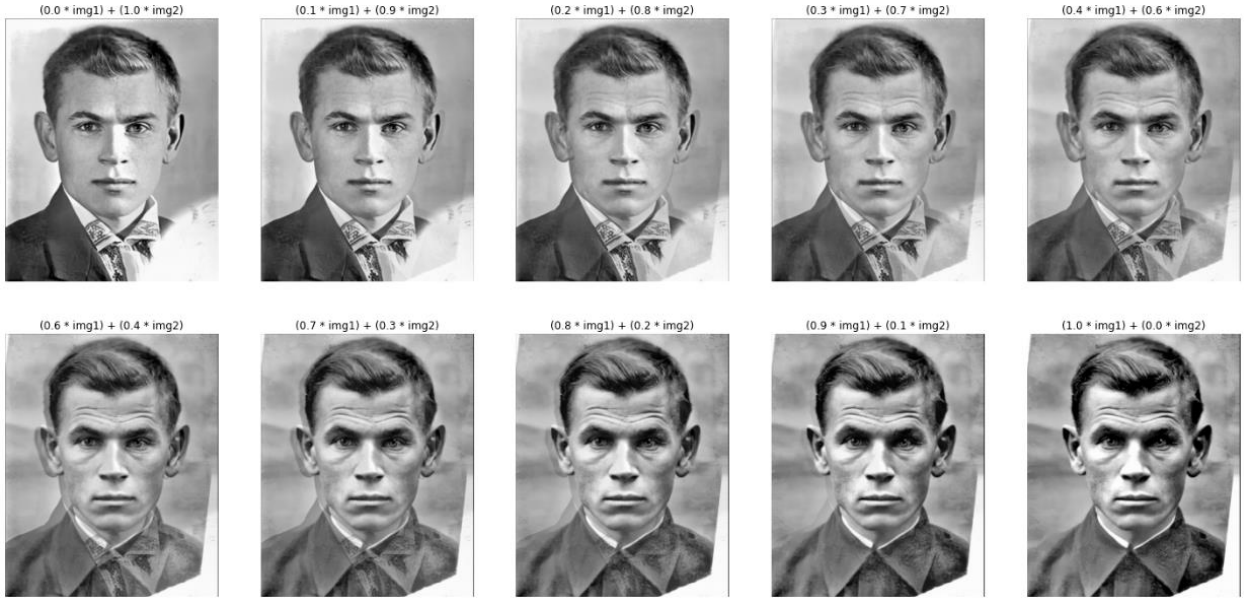
تصویر 1:

تصویر 2:

سوال 4)

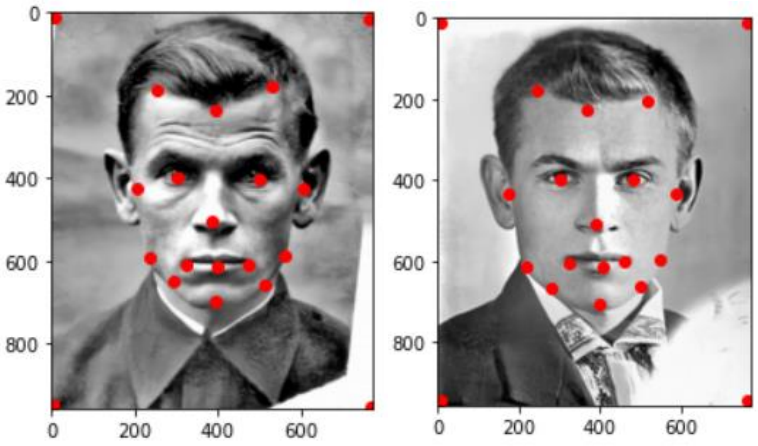
A) با استفاده از وزن دهی به دو عکس عکس 1 را در 10 مرحله به عکس دو تبدیل میکنیم.

```python
def naive_method(img1, img2, cnt):
    results = []
    names = []
    for alpha in np.linspace(0, 1, cnt):
        alpha = round(alpha, 1)
        naive_res = cv2.addWeighted(img1, 1 - alpha, img2, alpha, 0)
        results.append(naive_res)
        names.append('({} * img1) + ({} * img2)'.format(alpha, round(1 - alpha, 1)))
    return results, names
```

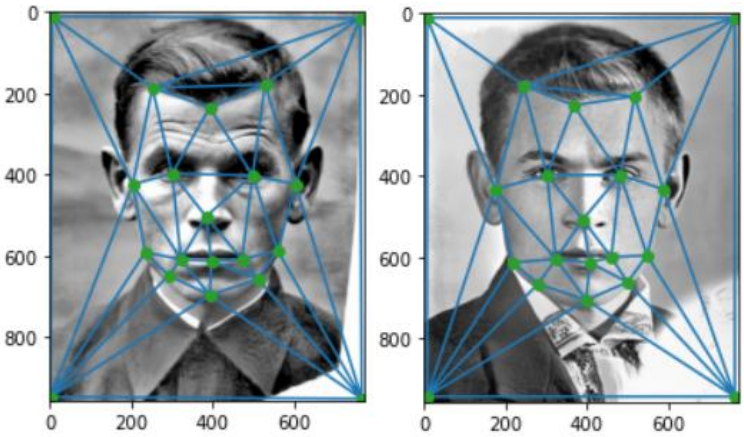


(B

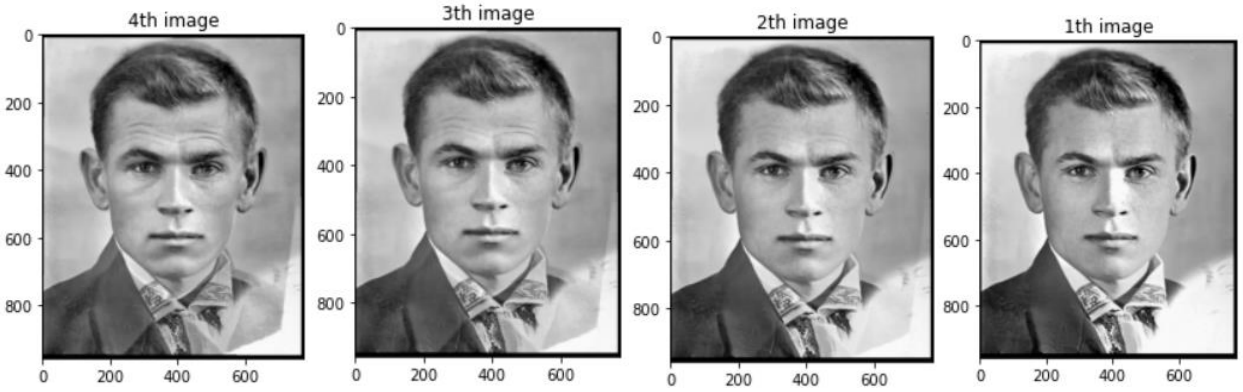
با استفاده از ginput 20 نقطه مهم در تصویر انتخاب میکنیم.



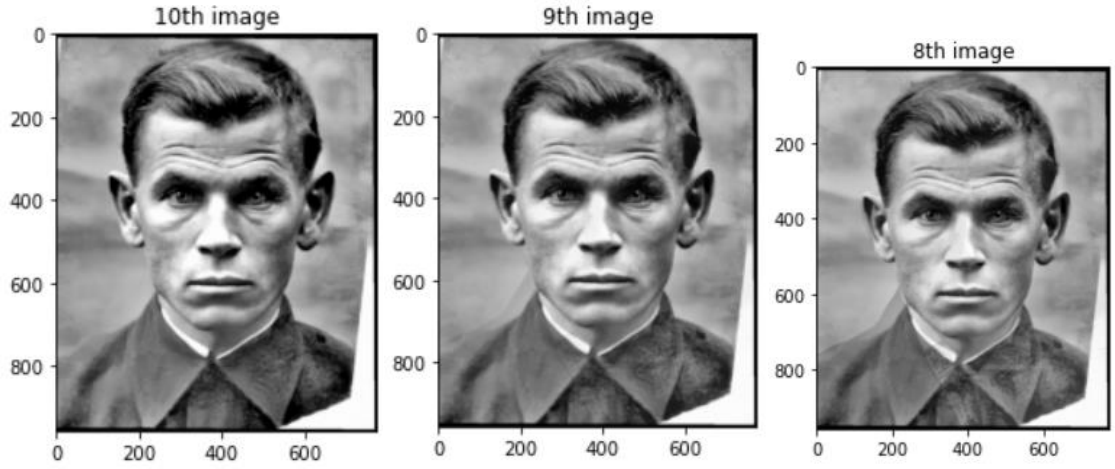
(C



(D


4th image
3th image
2th image
1th image


7th image
6th image
5th image



d)

این قسمت در فایل آورده شده است.

سوال 5)

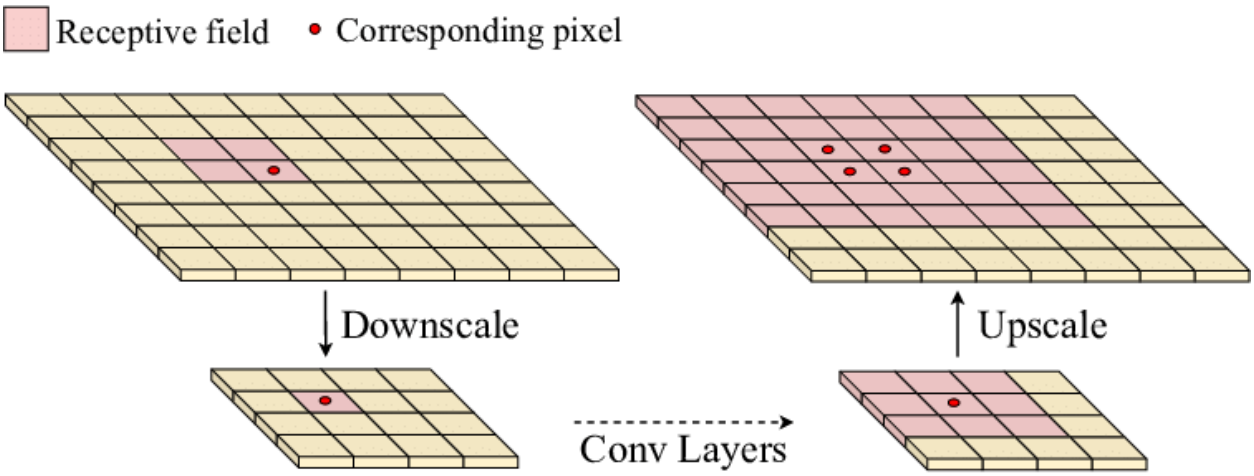
B)

در سوال آمده است که چرا اخرین ستون شامل دو 0 و یک 1 میباشد که با توجه به صورت سوال فکر میکنم منظور آخرین سطر بوده است که دارای دو 0 و یک 1 میباشد.

برای اطمینان از اینکه نقطه ای در صفحه z=1 به نقطه دیگری در همان صفحه نگاشت شده است. اگر بخواهید ردیف پایین را تغییر دهید، معمولاً با نقطه‌ای خارج از صفحه z=1 مواجه می‌شوید، و احتمالاً نمی‌دانید چگونه آن را به نقطه‌ای در صفحه دو بعدی معمولی برگردانید.

C)

بله با upscale و downscale های متناوب در نهایت عکس به شکل مشخص همگرا میشود



با توجه به شکل بالا میتوان ردیافت زمان که این عمل زیاد تکرار شود در نهایت عکسی بدست خواهد آمد که تنها یک رنگ مشخص دارد که شامل عملیات downscale , upscale خواهد بود

D)

خیر

هنگام تلاش برای اعمال یک تبدیل هندسی (یکی از فرآیندهای Crop، DynamicCrop، FastRotation، IntegerResample، Resample، و Rotation) به یک تصویر حاوی پیش نمایش، هشدار زیر نمایش داده می شود:

"موارد زیر در نتیجه تغییر هندسی حذف خواهند شد: پیش نمایش ها برخی از این عوارض جانبی ممکن است برگشت ناپذیر باشند. ادامه دهید؟"

این برای فرآیندهای DynamicCrop، Rotation، Resample و IntegerResample درست است.

همچنین در صفحه 21 کتاب quantum image processing نیز به این موضوع اشاره شده است.

E)

نویز های تناوبی ، نویز های هستند که به تصویر قابل اضاقه شدن هستند (additive) مثل اضافه کردن یک تابع سین میماند که به تصویر اضافه شده. مثلا به فوریه عکس مربوطه نویز سینسوسی را اضافه کنیم که یک نقطه روشن میشود. همچنین نویز های تناوبی با استفاده از band-rejection, notch filter نیز حذف میشوند.

همچنین نویز های نمک و فلفل نیز Additive هستند که به uniform noise نیز مشهور میبشند. نویز نمک و فلفل با انتخاب کسری از تصویر برای خراب کردن ایجاد شد (در هر پیکسل یک عدد تصادفی در محدوده 0 تا

1 با این کسر مقایسه شد تا مشخص شود که آیا پیکسل خراب شود یا نه) و سپس پیکسل‌ها را به صورت تنظیم کنید. تا روشنایی های تصادفی در محدوده 0 تا 255 خراب شده است.