پاسخ سوال اول

پس از استخراج نقاط کلیدی از دو تصویر، نیاز است تا نقاط متناظر با یکدیگر مشخص شوند. برای این منظور، ابتدا برای هر نقطه ویژگی یک توصیفگر محاسبه میشود. سپس، دو به دوی توصیفگرها از دو تصویر مقایسه میشوند و مشابهترین توصیفگرها به عنوان نقاط متناظر انتخاب میشوند برای جلوگیری از تناظریابی اشتباه حد آستانهای بر روی میزان مشابهت گذاشته میشود پس از یافتن نقاط متناظر، باید تابع تبدیلی را بدست آورد که نقاط تصویر اول را به نقاط تصویر دوم نگاشت کنند برای این کار، ابتدا یک مدل برای تابع تبدیل انتخاب میشود و سپس پارامترهای آن بر اساس نقاط بدست آمده بهینه میشوند حال در تبدیل شباهت که این تبدیل شامل چرخش، انتقال و مقیاس است 4 درجه آزادی و حداقل 2نقطه لازم است تا این مقادیر محاسبه گردند به دلیل وجود خطا در مکانیابی دقیق نقاط کلیدی، میتوان با استفاده از تعداد بیشتری نقطه به پارامترهای دقیقتری دست یافت دادههای پرت روش حداقل مربعات خطا حساس به دادههای پرت است روش RANSACبرای بدست آوردن تابع تبدیل مقاوم نسبت به دادههای پرت استفاده میشود.

پاسخ سوال دوم

در این سوال از راهنمایی ها ولینک داخل داک استفاده کردم. ابتدا لیستی از تصاویر ساختم و سپس به توابع داده شده دادم آن لیست را.

```
def stitch(image1, image2):

    out_img = None

    #Write your code here

    images=[]
    images.append(image1)
    images.append(image2)
    stitcher = cv2.Stitcher_create()
    (status, out_img) = stitcher.stitch(images)

    return out_img
```

نتیجه به صورت زیر است:

Result

img1

img2

پاسخ سوال سوم

در این سوال با توجه به موارد تدریس شده، و گفته شده در داک ابتدا:

1. لندمارک های چهره را پیدا کردم 2
2. با استفاده از این لینک نقاط طرفین ماسک را پیدا کردم
3. چهار نقطه از لندمارکهای صورت و 4 نقطه از ماسک را به تابع ترنسفورم دادم تا ماتریس ترنسفورم حاصل شود
4. سپس ماسک را با ماتریس تبدیل به تابع بعدی دادم تا تصویر ترنسفورم شده ماسک حاصل شود .
5. در انتها در نقاطی که ماسک مقدار داشت آن را به جای تصویر اصلی قرار داده ام.

نتیجه به صورت زیر بود:

result

face

```python
def put_mask(face, mask):
    result = face.copy()

    #Write your code here
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
    images = imutils.resize(face, width = 700)
    gray = cv2.cvtColor(images, cv2.COLOR_BGR2GRAY)
    rects = detector(gray, 1)

    for (i, rect) in enumerate(rects):
      shape = predictor(gray, rect)
      shape = face_utils.shape_to_np(shape)
      (x, y, w, h) = face_utils.rect_to_bb(rect)


    pts1 =np.float32([shape[3],shape[28],shape[13],shape[8]])
    pts2 =np.float32([[215,363],[613,106],[1013,355],[607,699]])

    M = cv2.getPerspectiveTransform(pts2,pts1)
    dst = cv2.warpPerspective(mask,M,(700,700))


    for i in range(face.shape[0]):
      for j in range(face.shape[1]):
        for k in range(face.shape[2]):
          if dst[i,j,k]>0:
             result[i,j,k]=dst[i,j,k]


    return result
```