سوال اول) Panorama with OpenCV:

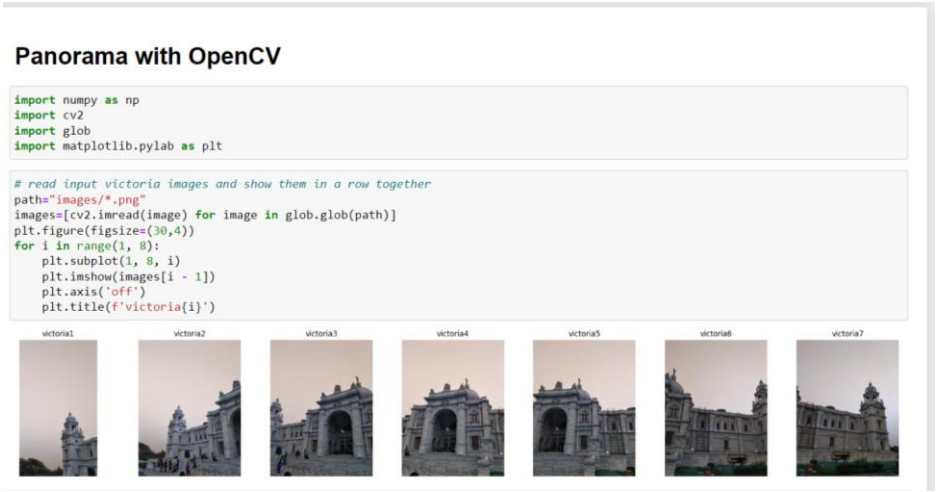
طبق مراحل خواسته شده در سوال جلو میرویم. ابتدا تصاویر برج را از فولدر داده شده میخوانیم و سپس آن ها را در یک ردیف در کنار هم به کمک plt نمایش میدهیم.

استفاده از لینک زیر برای نوشتن کد این قسمت:

https://stackoverflow.com/questions/71650269/how-to-display-multiple-images-at-once-with-matplotlib-in-one-figure

هم ی تصاویر برج دارای ئسوند png هستند برای همین آدرس فایل را ئوشه ی image و همه ی تصاویر دارای ئسوند png. دادم.

بعد یک لیست از تصاویر خوانده شده توسط cv2 ساختم. در نهایت آن ها را در یک ردیف 7 تایی نمایش دادم.



بعد از این مرحله از Stitcher_create استفاده میکنیم تا یک آبجکت پانوراما بسازیم برای این کار باید ورودی آن را 0 بدهیم. اگر status 0 باشد یعنی عملیات تشکیل تصویر پانوراما موفقیت آمیز بوده است. برای همین این شرط را چک میکنیم و اگر برابر با صفر بود تصویر تشکیل شده را نمایش میدهیم.

لینک استفاده شده: https://docs.opencv.org/4.x/d2/d8d/classcv_1_1Stitcher.html

سوال دوم) فرمول مربوط به تخمین زاویه چرخش میان دو تصویر:

در صفحه ی ۲۸ از جلسه ی ۱۲ام فرمول های بررسی شده مربوط به انتقال تصویر هستند حالا ما باید روابط گفته شده را برای چرخش تصویر به اندازه ی زاویه ی θ بنویسیم و به جای t_x و t_y اندازه ی زاویه را حساب کنیم.

حداقل مربعات خطا

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- تابع هزینه

$$cost = \sum (x_2^n - x_1^n - t_x)^2 + \left(y_2^n - y_1^n - t_y\right)^2$$

- بهینه‌سازی
  - محاسبه مشتق

$$\frac{d}{dt_x} cost = -2 \sum (x_2^n - x_1^n - t_x) = 0$$

$$\Rightarrow t_x = \frac{1}{N} \sum (x_2^n - x_1^n) \qquad t_y = \frac{1}{N} \sum (y_2^n - y_1^n)$$

برای اینکه تبدیل دچار خطا نشود از تمام نقاط تبدیل استفاده میکنیم و مانند اسلاید بالا برای اندازه‌گیری خطا می توانیم از حداقل مربعات و بهینه سازی استفاده کنیم.

حداقل مربعات:

از معادله ی صورت سوال داریم:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1 cos\theta - y_1 sin\theta \\ x_1 sin\theta + y_1 cos\theta \end{bmatrix} => \; x_2 = x_1 cos\theta - y_1 sin\theta \; , y_2 = x_1 sin\theta + y_1 cos\theta$$

X2, y2 مقادیر ثانویه ی *x1* و *y1* هستند که باید تفاضلشان را با هم بدست آورده و درفرمول *MSE* قرار بدهیم.

$$x_2 - x_1 cos\theta + y_1 sin\theta$$

$$y_2 - x_1 sin\theta - y_1 cos\theta$$

چون *n* نقطه داریم میانگین مربعات خطا را برای *n* دقطه به دست می آوریم:

$$cost = \frac{1}{n} \sum_{i=1}^{n} (x_2^n - x_1^n \cos\theta + y_1^n \sin\theta)^2 + (y_2^n - x_1^n \sin\theta - y_1^n \cos\theta)^2$$

سپس از این تابع نسبت به θ مشتق میگیریم:

$$\frac{d}{d\theta}cost = 2\sum_{i=0}^{n}(x_1^n \sin\theta + y_1^n \cos\theta)(x_2^n - x_1^n \cos\theta + y_1^n \sin\theta) + (-x_1^n \cos\theta + y_1^n \sin\theta)(y_2^n - x_1^n \sin\theta - y_1^n \cos\theta)$$

حالا باید θ هایی که مشتق را صفر میکنند پیدا کنیم:

$$\frac{d}{d\theta}cost = 2\sum_{i=0}^{n}(x_2^n\, x_1^n \sin\theta - (x_1^n)^2 \sin\theta\cos\theta + x_1^n y_1^n (\sin\theta)^2 + x_2^n\, y_1^n \cos\theta - x_1^n y_1^n (\cos\theta)^2 + (y_1^n)^2 \sin\theta\cos\theta) + (-x_1^n\, y_2^n \cos\theta + (x_1^n)^2 \sin\theta\cos\theta + x_1^n y_1^n (\cos\theta)^2 + y_2^n\, y_1^n \sin\theta - x_1^n y_1^n (\sin\theta)^2 - (y_1^n)^2 \sin\theta\cos\theta) = 0$$

$$\rightarrow \sum_{i=0}^{n} \sin\theta\,(x_2^n\, x_1^n + y_2^n\, y_1^n) + \cos\theta\,(x_2^n\, y_1^n - x_1^n\, y_2^n) = 0$$

$$\rightarrow \sin\theta \sum_{i=0}^{n}(x_2^n\, x_1^n + y_2^n\, y_1^n) + \cos\theta \sum_{i=0}^{n}(x_2^n\, y_1^n - x_1^n\, y_2^n) = 0$$

$$\rightarrow \tan\theta = \frac{\sum_{i=0}^{n}(x_1^n\, y_2^n - x_2^n\, y_1^n)}{\sum_{i=0}^{n}(x_2^n\, x_1^n + y_2^n\, y_1^n)} \rightarrow \theta = arctan\left(\frac{\sum_{i=0}^{n}(x_1^n\, y_2^n - x_2^n\, y_1^n)}{\sum_{i=0}^{n}(x_2^n\, x_1^n + y_2^n\, y_1^n)}\right)$$

منبع: پاورپینت و صحبت های استاد در جلسه ی ۱۲و ۱۳

سوال سوم) طراحی CamScanner:

الف)نگاشت GrayScale:

```
def to_grayscale(im):
    # Your code goes here.
    img = im.copy()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return gray
```

```
grayscale = to_grayscale(im)
imshow(grayscale)
```

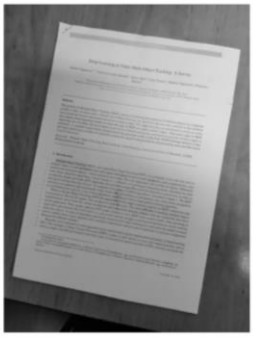
در تابع مشخص شده تصویر کپی از تصویر ورودی تولید کردیم رنگ آن را به سیاه-سفید تغییر دادیم و ریترن کردیم.

ب)محو کردن تصویر:

در فایل نوت بوک این قسمت سوال گفته شده است از دو نوع فیلتر گوسی و بایلیترال میتوانیم استفاده کنیم که من برای محو کردن از فیلتر بایلیترال استفاده کردم و آرگومان های **d, sigmaColor, sigmaSpace** را به ترتیب ۱۵و ۱۵و۹ قرار دادم.

```
def blur(im):
    # Your code goes here.
    bilateral = cv2.bilateralFilter(im, 9, 15, 15)
    return bilateral
```

```
blurred = blur(grayscale)
imshow(blurred)
```



پ)تشخیص لبه ها:

با استفاده از **Canny** تابع مورد سوال را کامل کردم.

```
def to_edges(im):
    # Your code goes here.
    edges = cv2.Canny(im,14,150,apertureSize = 3)
    return edges
```

```
edges = to_edges(blurred)
imshow(edges)
```

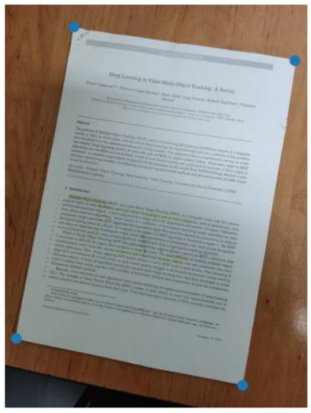
ت)تشخیص رئوس برگه با شناسایی contour های برگه:

کد این بخش را هم در تابع قرار داده شده نوشتم. با استفاده از **findContour**،**contour** های تصویر را پیدا کردیم سپس تغییرات لازم را برای تبدیل آنها به نوع نقطه ای مورد نظر از طریق تابع **approxPolyDP** اعمال کردیم. آرایه ی نقاط را به **np.ndarray** تبدیل کردیم.

```python
def find_vertices(im):
    # Your code goes here.
    contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    simplified_contours = []

    for cnt in contours:
        hull = cv2.convexHull(cnt)
        simplified_contours.append(cv2.approxPolyDP(hull, 0.001*cv2.arcLength(hull,True),True))
    simplified_contourss = np.array(simplified_contours, dtype = 'object')
    min_area = im.size
    biggest = None
    max_area = 0
    biggest_n = 0
    approx_contour=None
    for n,i in enumerate(simplified_contourss):
            area = cv2.contourArea(i)
            if area > min_area/10:
                    peri = cv2.arcLength(i,True)
                    approx = cv2.approxPolyDP(i,0.02*peri,True)
                    if area > max_area and len(approx)==4:
                            biggest = approx
                            max_area = area
                            biggest_n=n
                            approx_contour=approx
    dst = 0
    if approx_contour is not None and len(approx_contour)==4:
        approx_contour = np.float32(approx_contour)
        dst=approx_contour
    return dst.reshape((4,2))
```

```python
# Let's draw the points on the original image.
imshow(im)
vertices = find_vertices(edges)
plt.scatter([x for x, y in vertices], [y for x, y in vertices])
```

```
<matplotlib.collections.PathCollection at 0x1d9f7ba3d60>
```

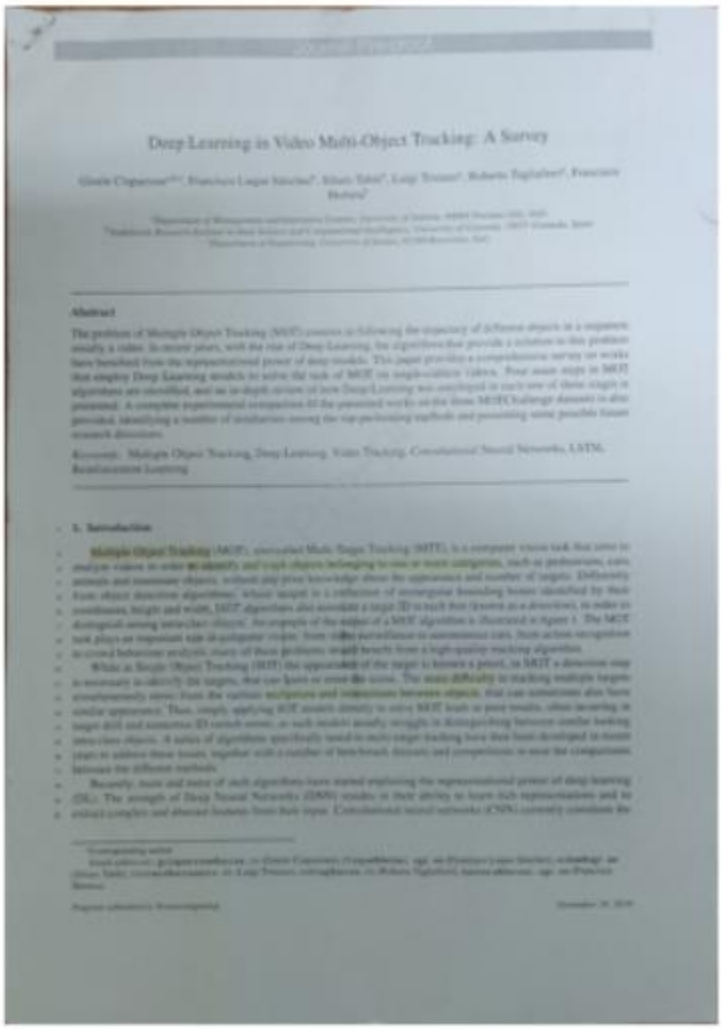


ث)نگاشت دورنما و برش:

با استفاده از نقاط پیدا شده هر دو طول کاغذ را حساب میکنیم و هر دو عرض کاغذ را هم حساب میکنیم. سپس بیشترین مقدار بین طول ها را و بیشترین عرض بین عرض ها را انتخاب میکنیم تا تصویر آسیب کمتری ببینند. سپس یک آرایه ی جدید از نقاط جدید میسازیم و به عنوان تارگت به تابعی که سوال گفته میدهیم تا خروجی مورد نظر تولید شود.

```python
def crop_out(im, vertices):
    # Your code goes here.

    vertices_reorser = reorder(vertices)

    (w1, w2, h1, h2) = vertices_reorser

    FirstWidth = np.sqrt((((w2[1] - w1[1]) ** 2) + (w2[0] - w1[0]) ** 2))

    SecondWidth = np.sqrt((((h1[1] - h2[1]) ** 2) + (h1[0] - h2[0]) ** 2))

    FirstHeight = np.sqrt((((w2[1] - h1[1]) ** 2) + (w2[0] - h1[0]) ** 2))

    SecondHeight = np.sqrt((((w1[1] - h2[1]) ** 2) + (w1[0] - h2[0]) ** 2))

    Best_Width = max(int(FirstWidth), int(SecondWidth))

    Best_Height = max(int(FirstHeight), int(SecondHeight))

    a = [0, 0]
    b = [Best_Width - 1, 0]
    c = [Best_Width - 1, Best_Height - 1]
    d = [0, Best_Height - 1]
    dst = np.array([ a, b, c, d], dtype = "float32")

    transform = cv2.getPerspectiveTransform(vertices_reorser, dst)  # get the top or bird eye view effect

    return cv2.warpPerspective(im, transform, (Best_Width, Best_Height))
```

```
cropped = crop_out(im, vertices)
imshow(cropped)
```

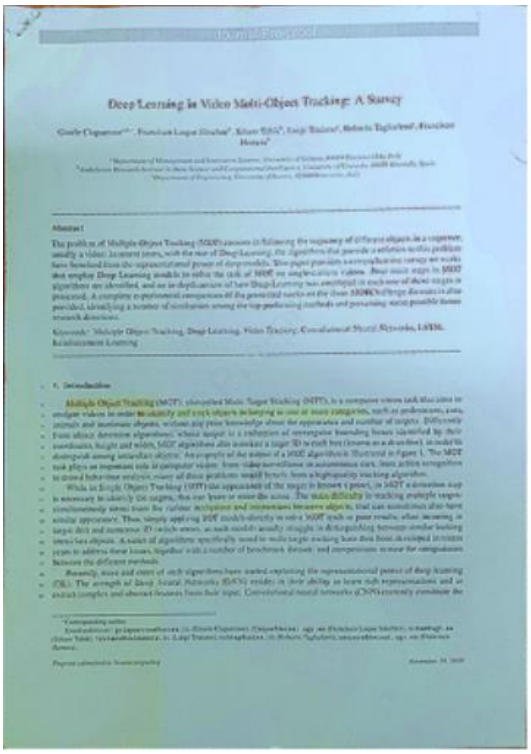


ج)بهبود تصویر:

```python
def enhance(im):
    # Your code goes here.
    kernel_sharpening = np.array([[0,-1,0],
                                  [-1, 5,-1],
                                  [0,-1,0]])
    sharpened = cv2.filter2D(im, -1, kernel_sharpening)
    HueSaturationValue = cv2.cvtColor(sharpened, cv2.COLOR_BGR2HSV)
    hue, Saturation, Value = cv2.split(HueSaturationValue)
    lim = 255 - 30
    Value[Value > lim] = 255
    Value[Value <= lim] += 30

    lim2 = 255 - 35
    Saturation[Saturation > lim] = 255
    Saturation[Saturation <= lim] += 35

    final_hsv = cv2.merge((hue, Saturation, Value))
    sharpened = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    return sharpened
```

ابتدا کرنل مورد نظر را تعریف کردیم. سپس با فیلتر دو بعدی کرنل شارپ کردن را روی تصویر اعمال کردیم. سپس hsv تصویر را بدست آوردیم. Value ها را بر اساس مقدار ۳۰ بهبود دادیم و saturation ها را براساس مقدار ۳۵ بهبود بخشیدیم.

نتیجه به صورت زیر شد:



پاسخ تمرین هفتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

لینک های استفاده شده:

https://www.tutorialspoint.com/how-to-perform-bilateral-filter-operation-on-an-image-in-opencv-using-python

https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

https://pythonexamples.org/python-opencv-cv2-find-contours-in-image/

https://stackoverflow.com/questions/8535650/how-to-change-saturation-values-with-opencv

https://levelup.gitconnected.com/create-your-own-camscanner-using-python-opencv-66251212270