

مبانی بینایی کامپیووتر

تمرین هفتم

نیکی نزاكتی

98522094

.1

برای خواندن عکس ها، نام آن ها را در یک لیست قرار داده و با کمک تابع cv2.imread آن ها را می خوانیم.



همچنین برای اینکه ابعاد عکس بزرگ است آن ها را resize می کنیم وارد لیستی از تصاویر می کنیم. سپس به صورت زیر با کمک تابع image sticher از کتابخانه OpenCV، لیست تصاویر را ورودی داده و تصاویر را به یکدیگر متصل می کنیم.

```
stitchy=cv2.Stitcher.create()  
(_,output)=stitchy.stitch(victoria)
```

خروجی نهایی به صورت زیر است:



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \rightarrow \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta x_1 - \sin\theta y_1 \\ \sin\theta x_1 + \cos\theta y_1 \end{bmatrix} \quad (2)$$

$$\text{Cost} = \sum_{i=1}^n (x_2^n - \cos\theta x_i + \sin\theta y_i)^2 + (y_2^n - \sin\theta x_i - \cos\theta y_i)^2$$

$$\begin{aligned} \frac{d}{d\theta} \text{Cost} &= \sum \frac{d}{dt} (x_2^n - \cos\theta x_i + \sin\theta y_i)^2 + \sum \frac{d}{dt} (y_2^n - \sin\theta x_i - \cos\theta y_i)^2 \\ &= \sum 2(x_i \sin\theta + y_i \cos\theta) + \sum 2(y_i \sin\theta - x_i \cos\theta) = 0 \end{aligned}$$

$$\Rightarrow \sum x_i^n \sin\theta + y_i^n \cos\theta = \sum x_i^n \cos\theta - y_i^n \sin\theta$$

$$\sin\theta \sum x_i^n + \cos\theta \sum y_i^n = \cos\theta \sum x_i^n - \sin\theta \sum y_i^n$$

$$\Rightarrow \sum x_i^n (\cos\theta - \sin\theta) = \sum y_i^n (\cos\theta + \sin\theta)$$

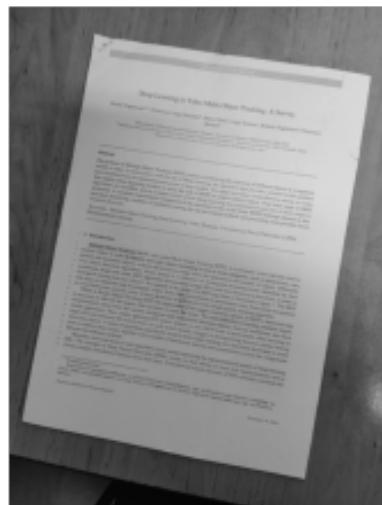
$$\Rightarrow \frac{\cos\theta + \sin\theta}{\cos\theta - \sin\theta} = \sum \frac{x_i^n}{y_i^n} \Rightarrow \frac{1}{\cos 2\theta + \tan 2\theta} = \sum \frac{x_i^n}{y_i^n}$$



.3

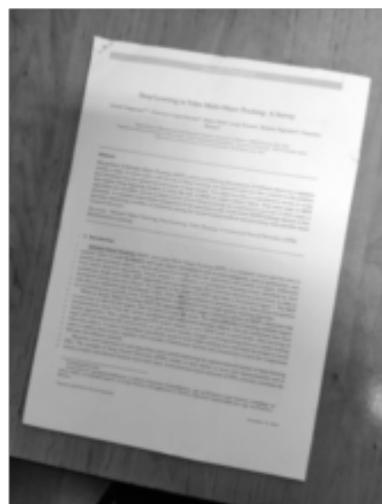
الف) ابتدا به صورت زیر تصویر را سیاه و سفید می کنیم.

```
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```



ب) سپس برای این که مولفه های فرکانس بالای تصویر حذف شوند تا در مراحل آتی کار، به ویژه تشخیص لبه ها ساده تر انجام شود، تصویر را محو می کنیم. برای این منظور از کرنل با سایز (5, 5) و سیگما 2 استفاده میکنیم.

```
blur = cv2.GaussianBlur(im, (5, 5), 2)
```



پ) با استفاده از canny لبه یابی میکنیم. حد پایین را ۳۰ و حد بالا را ۵۰ میگذاریم.
edge=cv2.Canny(im,30,50)



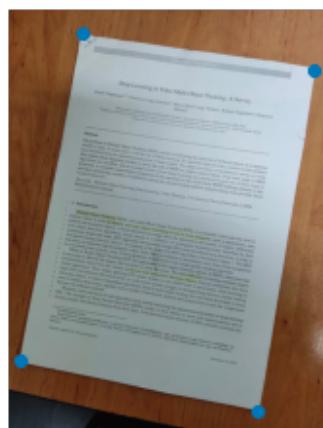
ت) برای تشخیص رئوس برگه، از تابع cv2.findContours استفاده میکنیم که نقاط پیوسته (در امتداد لبه ها) که رنگ یا شدت یکسانی دارند را به هم متصل میکند. همچنین از cv2.RETR_EXTERNAL استفاده میکنیم که کانتور های شدید را تشخیص می دهد و از cv2.CHAIN_APPROX_SIMPLE هم استفاده میکنیم که تنها endpoint های ضروری برای کشیدن خطوط کانتور را باز میگرداند.

```
contours, hierarchy = cv2.findContours(im, cv2.RETR_EXTERNAL,  
cv2.CHAIN_APPROX_SIMPLE)
```

سپس از cv2.approxPolyDP و cv2.convexHull استفاده میکنیم تا بزرگترین کانتور مستطیلی را پیدا کنیم.

```
hull = cv2.convexHull(c)  
points.append(cv2.approxPolyDP(hull, 0.001*cv2.arcLength(hull,True),True))
```

در ادامه از بین کانتور های بدست آمده بزرگترین را پیدا میکنیم و نقاط راسی را استخراج میکنیم.



ث) در این مرحله، مستطیلی که در مرحله قبل پیدا کرده بودیم را روی عکس اولیه تطبیق میدهیم، برای اینکار طول و عرض مستطیل را بدست آورده،

```
A_w= np.sqrt(((yr[0] - yl[0]) ** 2) + ((yr[1] - yl[1]) ** 2))
```

```
B_w = np.sqrt(((xr[0] - xl[0]) ** 2) + ((xr[1] - xl[1]) ** 2))
```

```
width = max(int(A_w), int(B_w))
```

```
A_h = np.sqrt(((xr[0] - yr[0]) ** 2) + ((xr[1] - yr[1]) ** 2))
```

```
B_h = np.sqrt(((xl[0] - yl[0]) ** 2) + ((xl[1] - yl[1]) ** 2))
```

```
height = max(int(A_h), int(B_h))
```

مستطیل را در numpy array ساخته،

```
target = np.array([[0, 0],[width - 1, 0], [width - 1, height - 1],[0, height - 1]], dtype = "float32")
```

و با استفاده از تابع cv2.getPerspectiveTransform تصویر اولیه را ترنسفرم کرده و با

cv2.warpPerspective آن را برش می دهیم.

```
transform = cv2.getPerspectiveTransform(vertices, target) # get the top or bird  
eye view effect
```

```
return cv2.warpPerspective(im, transform, (width, height))
```

خروجی:



ج) برای بهبود سازی تصویر، ابتدا فیلتر sharp کننده را روی آن اعمال می کنیم.

```
kernel = np.array([[0,-1,0],  
                  [-1, 5,-1],  
                  [0,-1,0]])  
sharpened = cv2.filter2D(im, -1, kernel)
```

سپس برای بهتر کردن روشنایی آن، تصویر را به HSV تبدیل میکنیم و مقدار V را با مقدار ۵ بهبود میدهیم و تصویر را به BGR باز میگردانیم.

```
hsv = cv2.cvtColor(sharpened, cv2.COLOR_BGR2HSV)  
h, s, v = cv2.split(hsv)  
lim = 255 - 50  
v[v > lim] = 255  
v[v <= lim] += 50  
final_hsv = cv2.merge((h, s, v))  
enhanced = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
```

خروجی نهایی camscanner ما به صورت زیر خواهد بود:



Resources:

<https://www.geeksforgeeks.org/>

<https://docs.opencv.org/>

<https://levelup.gitconnected.com/>