

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین سری هشتم
آشنایی با شار نوری

مهدی فیروزبخت
۴۰۰۱۳۱۰۲۷

استاد درس : دکتر رضا صفابخش

زمستان ۱۴۰۱
دانشگاه امیرکبیر
گروه هوش مصنوعی

سوال اول .

الگوریتم Shi-Thomas :

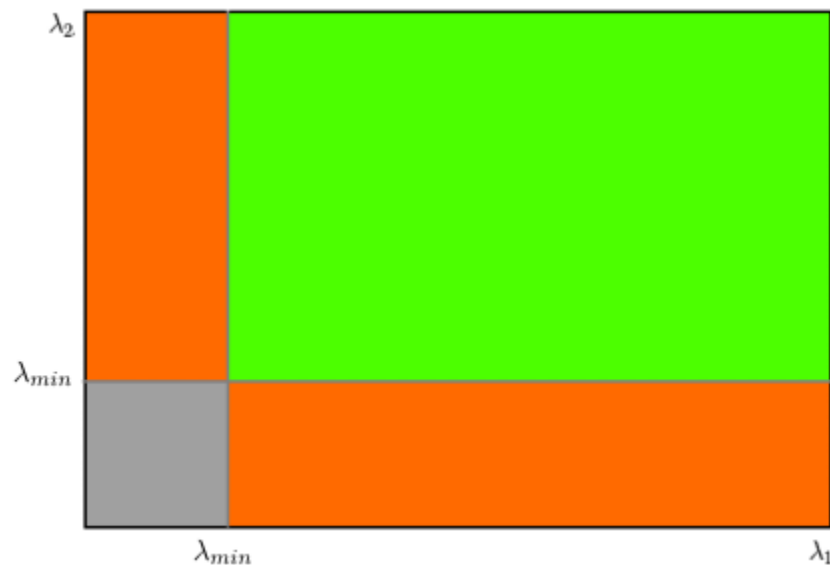
این الگوریتم شباهت زیادی به الگوریتم آشکارسازی گوشه هریس دارد. تابع امتیاز دهی در آشکارساز گوشه هریس به صورت زیر محاسبه میشود :

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

به جای این، شی توماسی پیشنهاد کرد:

$$R = \min(\lambda_1, \lambda_2)$$

اگر مقدار آن بیشتر از حد آستانه باشد، گوشه محسوب می شود. اگر آن را در فضای $\lambda_1 - \lambda_2$ ترسیم کنیم همانطور که در آشکارساز گوشه هریس انجام دادیم، تصویری مانند زیر دریافت می کنیم:



از شکل، می بینید که فقط زمانی که λ_1 و λ_2 بالاتر از مقدار حداقل، λ_{min} باشند، به عنوان یک گوشه (منطقه سبز) در نظر گرفته می شود.

OpenCV دارای تابعی به نام `cv2.goodFeaturesToTrack()` است. N قوی ترین گوشه تصویر را با روش Shi-Tomasi پیدا می کند. طبق معمول، تصویر باید یک تصویر خاکستری باشد. سپس تعداد گوشه هایی را که می خواهید پیدا کنید مشخص می کنید. سپس سطح کیفیت را مشخص می کنید که مقداری بین 0-1 است که نشان دهنده حداقل کیفیت گوشه ای است که زیر آن همه رد می شوند. سپس حداقل فاصله اقلیدسی بین گوشه های شناسایی شده را ارائه می کنیم.

با تمام این اطلاعات، تابع گوشه هایی را در تصویر پیدا می کند. تمام گوشه های زیر سطح کیفیت رد می شوند. سپس گوشه های باقی مانده را بر اساس کیفیت به ترتیب نزولی مرتب می کند. سپس تابع اولین گوشه قوی را می گیرد، تمام گوشه های نزدیک را در محدوده حداقل فاصله دور می اندازد و N قوی ترین گوشه را برمی گرداند.

پس بر اساس این اطلاعات ، تابع cv2.goodFeaturesToTrack در OpenCV برای تشخیص گوشه های یک تصویر استفاده می شود. دارای چندین پارامتر است که به شما امکان می دهد فرآیند تشخیص گوشه را سفارشی کنید:

۱. image: این تصویر ورودی است که می خواهید گوشه ها را در آن تشخیص دهید.
 ۲. maxCorners: این پارامتر حداکثر تعداد گوشه هایی را که می خواهید در تصویر شناسایی کنید را مشخص می کند. این تابع قوی ترین گوشه های maxCorners را در تصویر برمی گرداند.
 ۳. qualityLevel: این پارامتر سطح کیفی گوشه های شناسایی شده را مشخص می کند. مقدار 0.01 به این معنی است که تنها 1٪ از قوی ترین گوشه ها انتخاب شده اند، در حالی که مقدار 1.0 به این معنی است که همه گوشه ها انتخاب شده اند.
 ۴. minDistance: این پارامتر حداقل فاصله بین دو گوشه را مشخص می کند. اگر دو گوشه در این فاصله تشخیص داده شود، تنها قوی ترین آن انتخاب می شود.
 ۵. mask: این یک پارامتر اختیاری است که به شما امکان می دهد منطقه مورد علاقه (ROI) را در تصویر مشخص کنید که در آن تشخیص گوشه باید انجام شود.
 ۶. blockSize: این پارامتر اندازه همسایگی مورد استفاده برای محاسبه قدر گرادیان برای تشخیص گوشه را مشخص می کند.
 ۷. useHarrisDetector: این پارامتر مشخص می کند که آیا از آشکارساز گوشه هریس (در صورت تنظیم روی True) یا آشکارساز گوشه Shi-Tomasi (در صورت تنظیم روی False) استفاده شود.
 ۸. k: این پارامتر فقط در صورتی استفاده می شود که از آشکارساز گوشه هریس استفاده شده باشد و پارامتر آزاد آشکارساز هریس را مشخص کند.
- با تنظیم این پارامترها، می توانید مبادله بین تعداد گوشه های شناسایی شده و کیفیت آنها را کنترل کنید. در این پیاده سازی که در فایل Q1 قرار داده شده ابتدا یک ویدیو ایجاد میکنیم. سپس ویدیو را ذخیره میکنیم و از فریم اول آن استفاده میکنیم تا پارامتر های فریم اول ویدیو را برای تابع goodFeaturesToTrack به دست آوریم.
- ابتدا ویدیو با نام Q1 ایجاد میکنیم :

```
# This will return video from the first webcam on your computer.
cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
videoWriter = cv2.VideoWriter('Q1.avi', fourcc, 30.0, (640, 480))
```

سپس فریم اول را جدا سازی میکنیم.تصویر را به سطح خاکستری برده و با فیلتر گاسی آن را هموارسازی میکنیم:

```
ret, frame = cap.read()
old_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
old_gray = cv2.GaussianBlur(old_gray, ksize=(15, 15), sigmaX=2, sigmaY=2)
```

سپس پارامترهای تابع را در نظر گرفته و به تابع اعمال میکنیم. برای مشاهده کیفیت کار صورت گرفته دایره هایی در تصویر ایجاد میکنیم در نقاط تشخیص داده شده تا اعمال آن را به تصویر را دریابیم :

```
feature_params = dict(maxCorners=10,
                      qualityLevel=0.06,
                      minDistance=200,
                      blockSize=7)

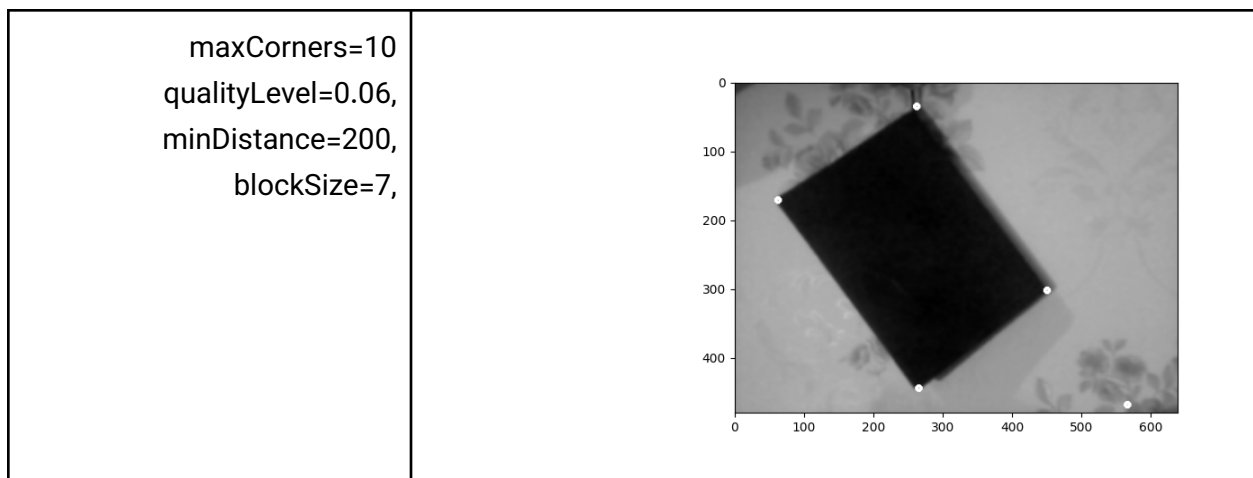
p1 = cv2.goodFeaturesToTrack(old_gray, mask=None, **feature_params)
# cv2.imshow("d", old_gray)

corners = np.intp(p1)

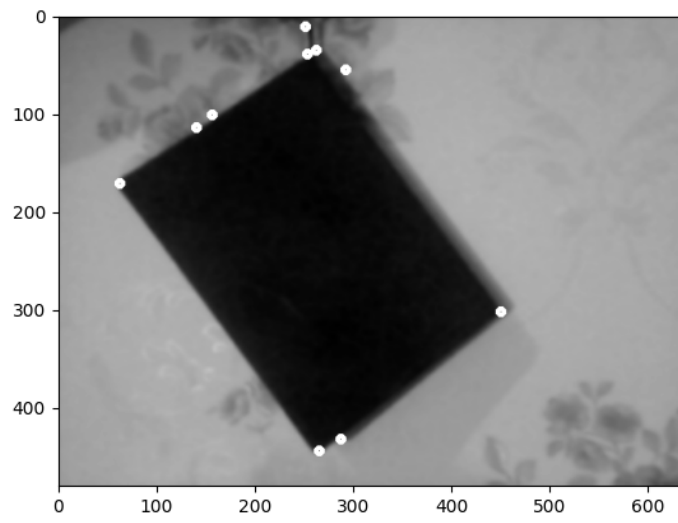
for i in corners:
    x, y = i.ravel()
    cv2.circle(old_gray, (x, y), 3, 255, -1)

plt.imshow(old_gray, cmap='gray')
plt.show()
```

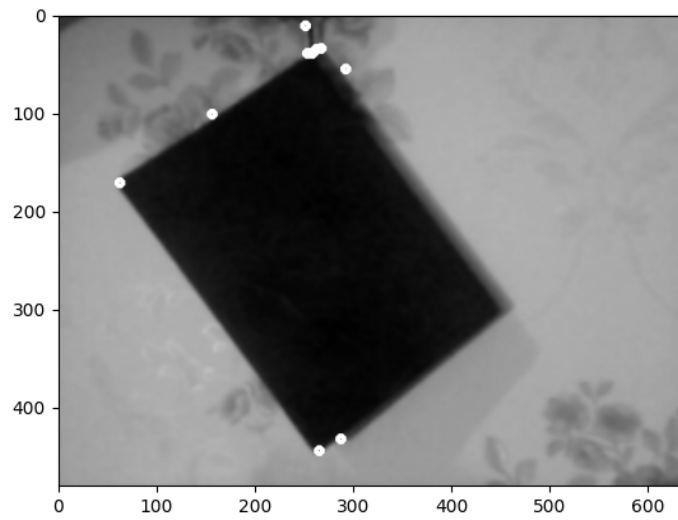
نتایج به دست آمده برای مقادیر متفاوت به صورت زیر میباشد :



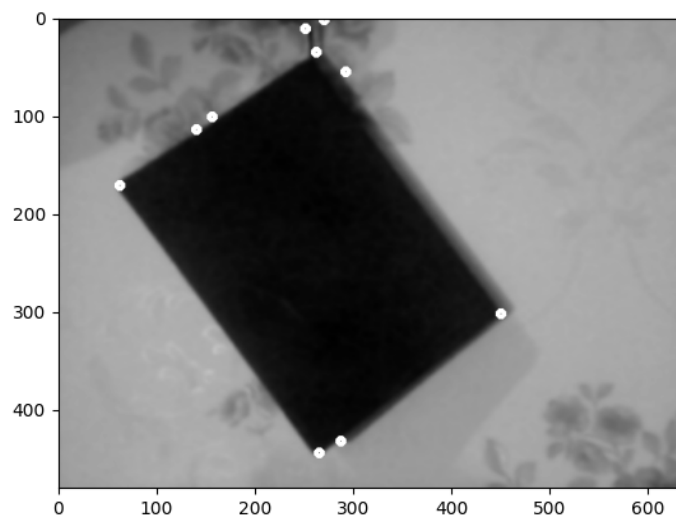
maxCorners=10
qualityLevel=0.01,
minDistance=7,
blockSize=7,



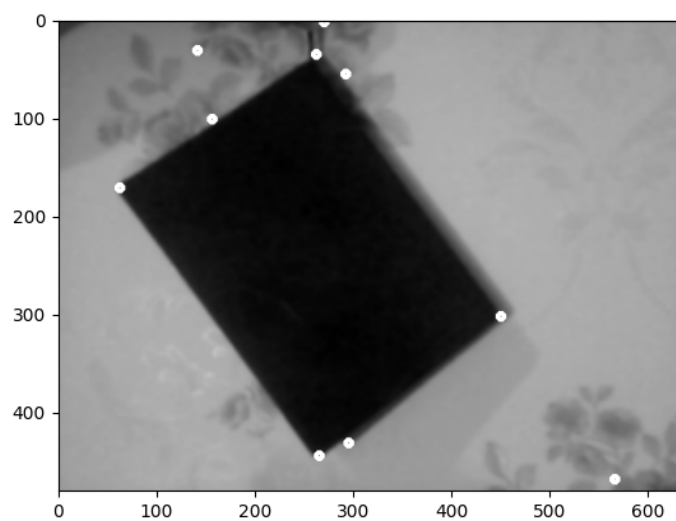
maxCorners=10
qualityLevel=0.01,
minDistance=1,
blockSize=7,



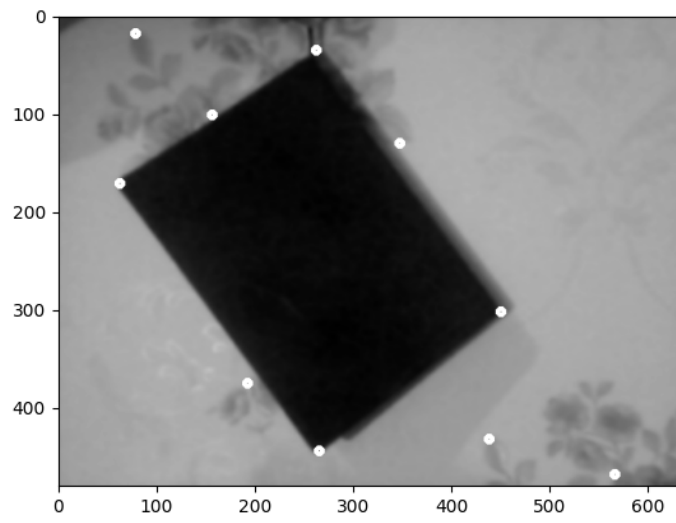
maxCorners=10
qualityLevel=0.01,
minDistance=10,
blockSize=7,



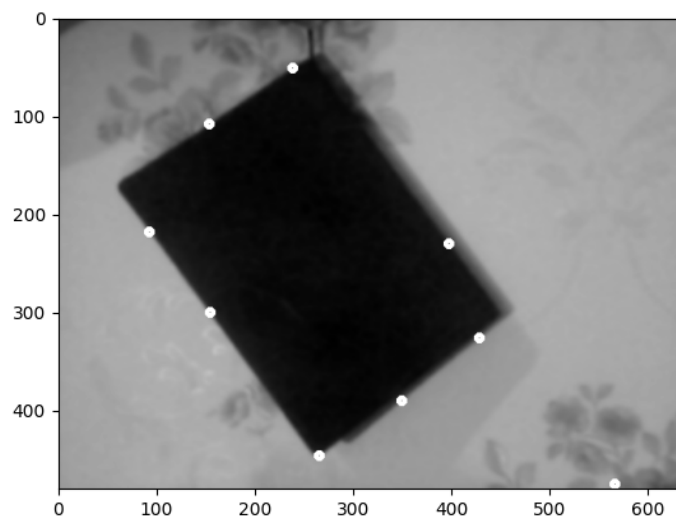
maxCorners=10
qualityLevel=0.01,
minDistance=30,
blockSize=7,



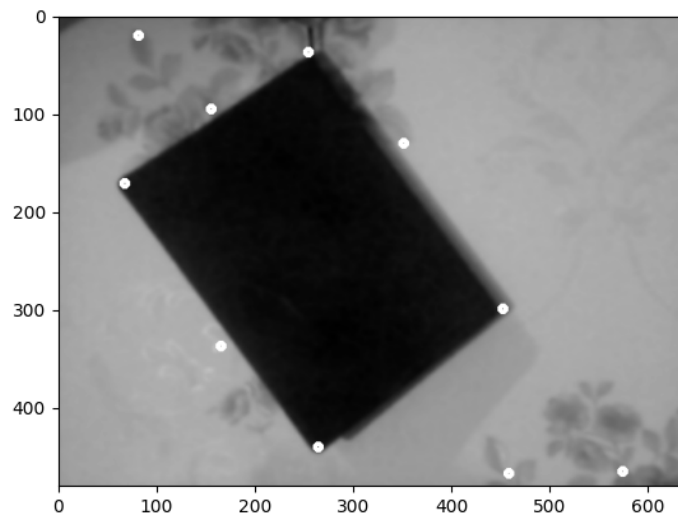
maxCorners=10
qualityLevel=0.01,
minDistance=100,
blockSize=7,



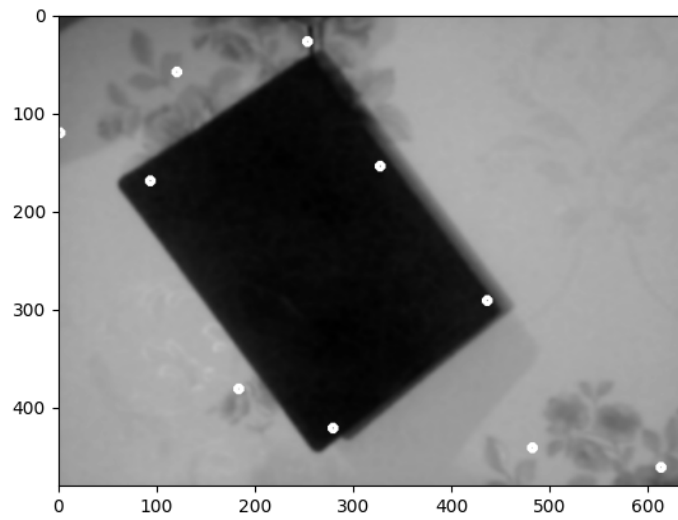
maxCorners=10
qualityLevel=0.01,
minDistance=100,
blockSize=1,



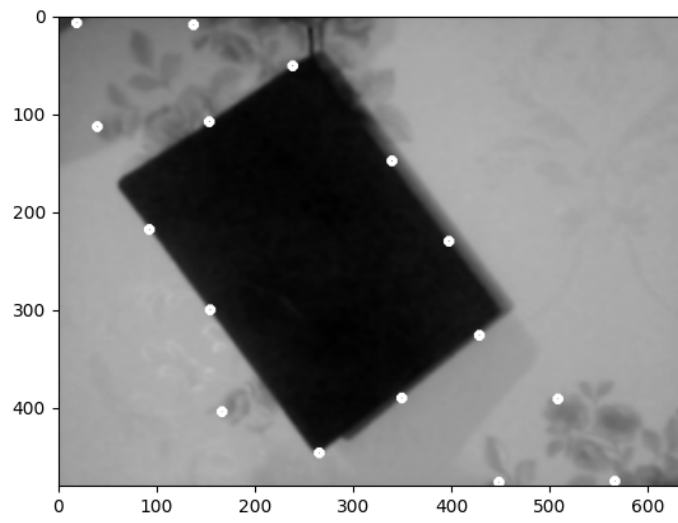
maxCorners=10
qualityLevel=0.01,
minDistance=100,
blockSize=20,



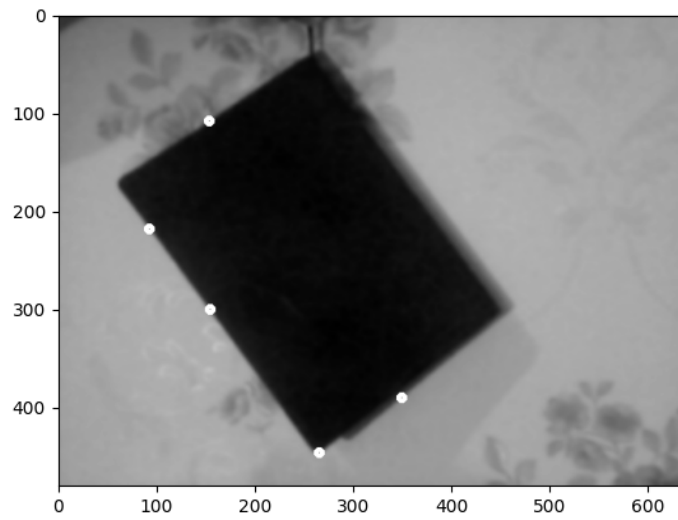
maxCorners=10
qualityLevel=0.01,
minDistance=100,
blockSize=100,



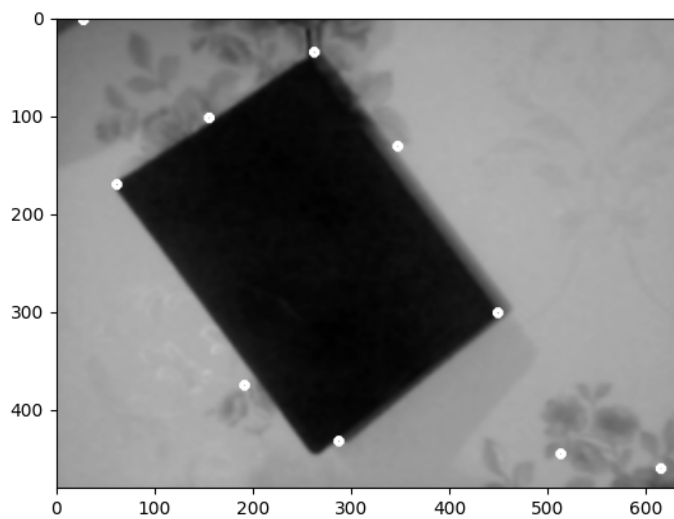
maxCorners=20
qualityLevel=0.01,
minDistance=100,
blockSize=1,



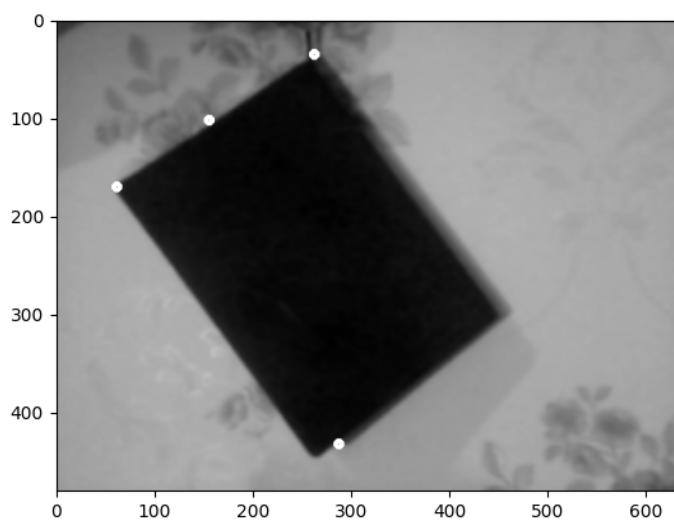
maxCorners=5
qualityLevel=0.01,
minDistance=100,
blockSize=1,

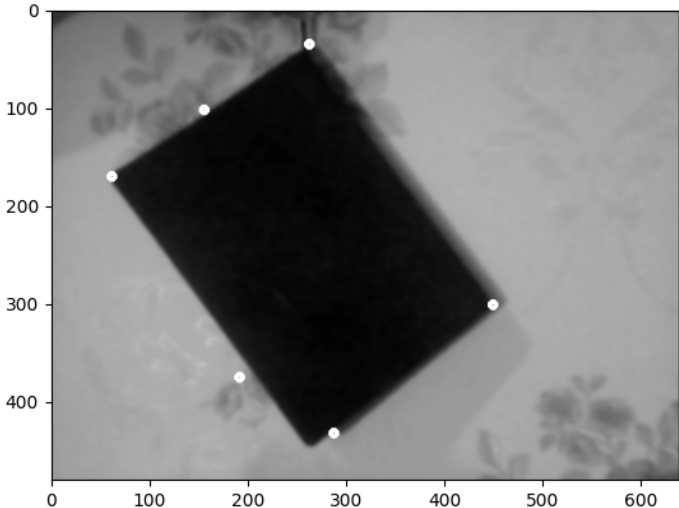
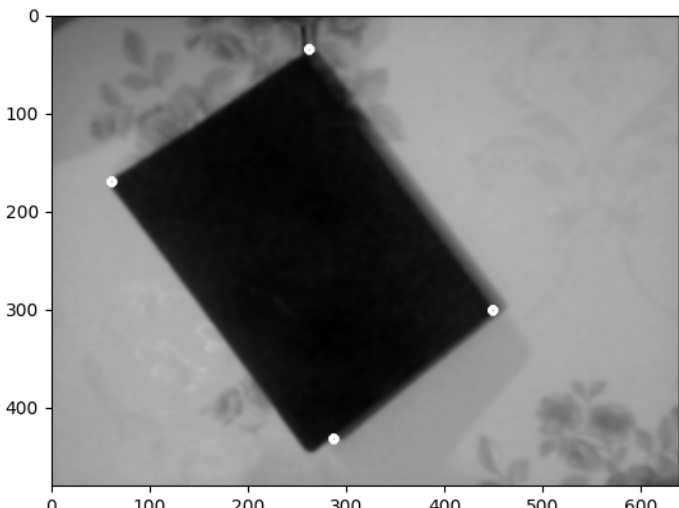


maxCorners=10
qualityLevel=0.01,
minDistance=100,
blockSize=5,



maxCorners=10
qualityLevel=0.15,
minDistance=100,
blockSize=5,



<pre> maxCorners=10 qualityLevel=0.135, minDistance=100, blockSize=5, </pre>	
<pre> maxCorners=10 qualityLevel=0.135, minDistance=120, blockSize=5, </pre>	

بهترین نتیجه به دست آمده برای این ویدیو و دوربین مقدار کرنر های ۱۰ ، سطح کیفیت نقاط ۰.۱۳۵، حداقل فاصله ۱۲۰ و اندازه بلاک ۵ میباشد.

ویدیو این تصویر در فایل Q1 میباشد.

۲.

برای این سوال نیازمند این هستیم در هر فریم مقدار تابع calcOpticalFlowPyrLK را به دست بیاوریم. به همین دلیل باید پارامترهای این تابع را نیز مشخص نماییم.

```
lk_params = dict(winSize=(20, 20),  
                 maxLevel=2,  
                 criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
```

سپس در هر فریم ابتدا مقدار تابع calcOpticalFlowPyrLK را میابیم. ورودی‌های این تابع، فریم قبلی، فریم فعلی و نقاط مهم فریم قبلی هستند به همراه پارامترهای مشخص شده.

خروجی‌های این تابع :

nextPts - بردار خروجی نقاط 2 بعدی حاوی موقعیت‌های جدید محاسبه شده ویژگی‌های ورودی در تصویر دوم.

status - بردار وضعیت خروجی (از کاراکترهای بدون علامت)؛ اگر جریان ویژگی‌های مربوطه پیدا شده باشد، هر عنصر بردار روی 1 تنظیم می‌شود، در غیر این صورت، 0 تنظیم می‌شود.

err - بردار خروجی خطاها. هر عنصر بردار برای ویژگی مربوطه روی یک خطا تنظیم می‌شود، نوع اندازه‌گیری خطا را می‌توان در پارامتر پرچم تنظیم کرد. اگر جریان پیدا نشد، خطا تعریف نشده است.

بر اساس این خروجی‌ها نقاط جدید در این فریم جدید پیدا می‌شود. اگر مقدار status برای حالتی برابر 1 بود در این موقعیتی وجود دارد.

حال موقعیت‌ها را برای 3 فریم ذخیره‌سازی می‌کنیم. Endpoints2 همیشه موقعیت نقاط جدید است. Endpoints1 موقعیت نقاط در فریم قبلی و endpoints0 موقعیت در 2 فریم قبلی است.

اگر موقعیت فریم قبلی نسبت به موقعیت فریم بعدی تغییر داشته است با بردار آبی نمایش داده می‌شود.

اگر موقعیت فریم قبلی نسبت به فریم قبلی آن تغییر داشته باشد با بردار سبز نمایش داده می‌شود.

ویدیو مربوط به این سوال در فایل Q2 قرار داده شده است.

```

ret, frame = cap.read()
frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
frame_gray = cv2.GaussianBlur(frame_gray, ksize=(15, 15), sigmaX=2, sigmaY=2)

p1, state, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p1, None, **lk_params)
old_gray = frame_gray.copy()
endpoints2 = []
if len(p1) > 0:
    for i, s in enumerate(state):
        if s == 1:
            endpoints2.append(p1[i])
        else:
            endpoints2.append(None)
if len(endpoints2) > 0 and len(endpoints1) > 0:
    for old, new in zip(endpoints1, endpoints2):
        if old is not None and new is not None:
            old = old[0].astype(int)
            new = new[0].astype(int)
            frame = cv2.arrowedLine(frame, old, new, (255, 0, 0), 3)
if len(endpoints1) > 0 and len(endpoints0) > 0:
    for old, new in zip(endpoints0, endpoints1):
        if old is not None and new is not None:
            old = old[0].astype(int)
            new = new[0].astype(int)
            frame = cv2.arrowedLine(frame, old, new, (0, 255, 0), 3)
endpoints0 = endpoints1.copy()
endpoints1 = endpoints2.copy()

```

۳.

در مورد این مدل از HSV پیروی میکند. یک هیت مپ از تصویر به ما میدهد. ابتدا یک ماسک HSV به اندازه تصویر ایجاد میکند که این ماسک تماما سیاه است. سپس مقدار sat آن را ۲۵۵ قرار میدهیم. سپس در هر فریم با استفاده از `calcOpticalFlowFarneback` جریان شار نوری را محاسبه میکنیم. سپس با تابع `cartToPolar` قدر و زاویه بردار دوبعدی را محاسبه میکنیم. در مرحله بعدی مقدار رنگ تصویر را با توجه به زاویه جریان نوری تنظیم میکنیم. همچنین مقدار را بر اساس مقدار نرمال شده جریان نوری تنظیم میکنیم. سپس به RGB تبدیل میکنیم.

```
while True:
    # Capture another frame and convert to gray scale
    _, frame2 = capture.read()
    next = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
    next = cv2.GaussianBlur(next, ksize=(15, 15), sigmaX=4, sigmaY=4)
    # Optical flow is now calculated
    flow = cv2.calcOpticalFlowFarneback(prvs, next, None, 0.5, 3, 15, 3, 5, 1.2, 0)
    # Compute magnite and angle of 2D vector
    mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    # Set image hue value according to the angle of optical flow
    hsv_mask[..., 0] = ang * 180 / np.pi / 2
    # Set value as per the normalized magnitude of optical flow
    hsv_mask[..., 2] = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)
    # Convert to rgb
    rgb_representation = cv2.cvtColor(hsv_mask, cv2.COLOR_HSV2BGR)

    cv2.imshow('frame2', rgb_representation)
    videoWriter.write(rgb_representation)
    kk = cv2.waitKey(30) & 0xff
    # Press 'e' to exit the video
    if time() - start_time >= 30:
        break
```

تصویر به دست آمده از این مدل در فایل Q3 ذخیره شده است.

در این مدل ها نیازمند این هستیم که توضیحات مشاهده شده را ارائه نماییم. مدل اول که به نقاط گوشه حساس است به علت اینکه نیازمند این است بر اساس هر ویدیو جدید تنظیم شود شاید نسبت به مدل دوم دارای سختی بیشتری باشد. در هنگام حرکت دادن آرام مشکلی وجود ندارد و نقاط به خوبی منتقل میشوند. این حالت برای مدل دوم نیز صادق است.

در هنگام حرکت سریع در مدل اول نقاط ممکن است گم شوند که علت این اتفاق این است که چون ممکن است در دو فریم متوالی جا به جایی به حدی زیاد باشد که قابل اندازه گیری برای مدل نباشد و به همین علت مدل دچار اشکال میشود. راهکار آن از نظر بنده این است سرعت تصویر برداری افزایش یابد تا در هر ثانیه فریم بیشتری دریافت شده و مدل بتواند جا به جایی را محاسبه کند. در مورد تصویر دوم نیز این مشکل با شدت کمتری اتفاق می افتد. به این صورت که تصویر به حالتی دارای نویز شدید شده و نقاطی از دست میرود اما باز هم بهتر از مدل اول است. برای این روش نیز راهکار ارائه شده ممکن است از پاسخ خوبی دهد. در مورد چرخش نیز شرایط مانند حرکت سریع است اما در مجموع مدل دوم بهتر عمل میکند. به طور کلی می توانیم توضیحات زیر را ارائه نماییم.

الگوریتمهای مبتنی بر نقاط کلیدی :

- توصیفگر مناسبی از شی در دسترس باشد؛
- جسم توسط اجسام دیگر پوشانده میشود؛
- محیط پیچیده و شامل اجسام زیاد است؛
- شرایط نوری محیط دائماً تغییر می کند؛
- تغییرات شدیدی در مقیاس یا چرخش شی داریم؛
- یا حرکتهای جسم بسیار سریع و بزرگ است، عملکرد بهتری دارند.

الگوریتمهای مبتنی بر شار نوری در شرایطی که:

- توصیفگر مناسبی از شی در دسترس نیست ؛
- قصد رهگیری اشیاء مختلفی داریم؛
- یا حرکت جسم به صورت دقیق مورد نظر باشد ، عملکرد بهتری دارند.

برای مسئله خاصی که در این تمرین هستند به نظر می رسد با توجه به توضیحات فوق استفاده از الگوریتم های مبتنی بر نقاط کلیدی مناسب تر باشد.