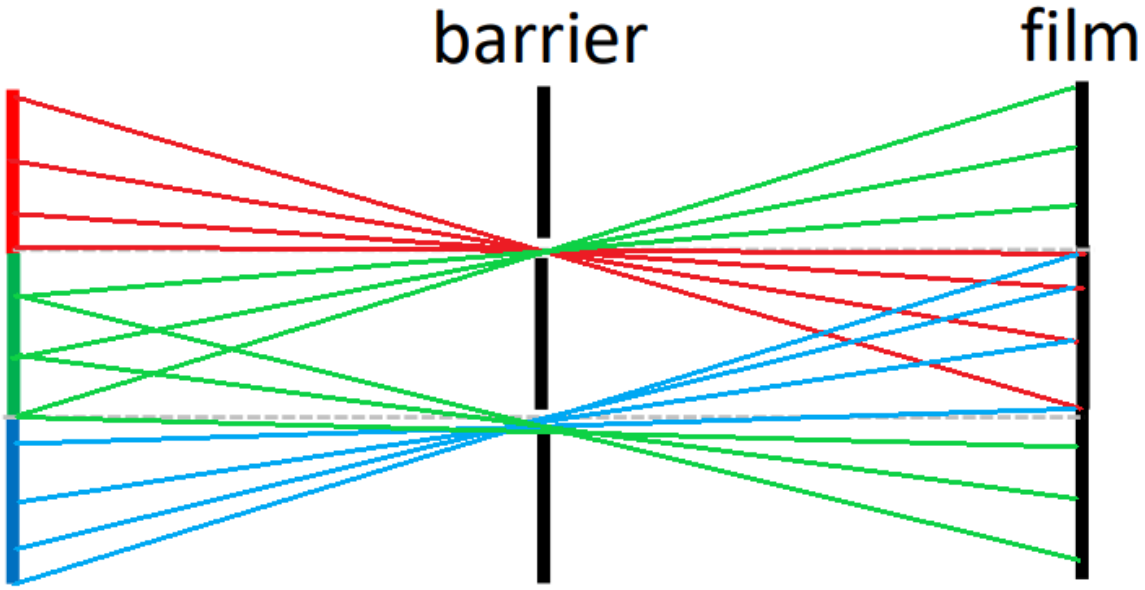
Computer Vision
Dr. Mohammadi
Fall 2022
Hoorieh Sabzevari - 98412004
HW2

۱. اگر دریچه را خیلی بزرگ کنیم باعث می‌شود نور منعکس شده در بخش بیشتری از تصویر تاثیر می‌گذارد و همچنین تصویر تار خواهد شد. اگر هم دریچه را خیلی کوچک کنیم، از تار شدن تصویر کاسته می‌شود اما مقدار نور واردشده به دوربین بسیار کم خواهد شد چون پرتوهای کمتری از دریچه می‌توانند عبور کنند. همچنین آن پرتوهای عبور یافته هم پراکنده خواهند شد.

برای قسمت دوم سوال، نورها به صورت زیر از دریچه‌ها رد می‌شوند. نور منعکس‌شده از بخش سبز به بخش‌های بالایی و پایینی فیلم می‌رسند اما نور منعکس‌شده از بخش قرمز و آبی به بخش وسط فیلم می‌رسند و چون هر پیکسل از بخش وسط فیلم توسط نور قرمز و آبی ثبت شده است لذا به رنگ ارغوانی ثبت خواهد شد.



۲. از دوربین لنزدار استفاده شده است. زیرا تاری فقط مربوط به یک ناحیه می‌باشد و اگر از دوربین pinhole استفاده شده بود باید کل تصویر تار می‌بود یا نور پراکنده می‌شد. همچنین به این فاصله از دوربین که اشیاء تقریبا واضح دیده می‌شوند، عمق میدان می‌گویند. در دوربین‌ها معمولا هم از لنز و هم از دریچه استفاده می‌شود و می‌توانیم عمق میدان را کنترل کنیم. همچنین با تغییر v(یعنی فاصله‌ی صفحه) تا لنز می‌توانیم u های متفاوتی (یعنی فاصله‌ی شی تا لنز) را ثبت کنیم.

۳. طبق فرضیات مسئله در ابتدا $v = 10cm$ ، $f = 10cm$ و $u = 30cm$ است.

f همان فاصله‌ی کانونی، u فاصله‌ی شی تا لنز و v فاصله‌ی لنز تا صفحه‌ی فیلم دوربین است.

همانطور که واضح است این سه پارامتر در فرمول زیر صدق نمی‌کنند.

طبق فرمول داریم:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

$$\rightarrow \frac{1}{10} = \frac{1}{30} + \frac{1}{v}$$

$$\rightarrow v = 15cm$$

که این به این معنی است که پرتوهای منعکس شده از یک پیکسل شی در نقطه‌ای دورتر از فیلم دوربین همگرا می‌شوند پس تصویر تار ثبت خواهد شد و کیفیت بدی خواهد داشت.

قسمت دوم سوال: $u + v = 40cm$

طبق فرمول داریم:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

$$\rightarrow \frac{1}{10} = \frac{1}{40 - v} + \frac{1}{v}$$

$$\rightarrow v = u = 20cm$$

لذا با ثابت ماندن فاصله‌ی بین دوربین و شی لنز را باید وسط این فاصله یعنی در ۲۰ سانتی‌متری دوربین قرار داد.

۱-۴. در عمل هیچ لنزی ایده‌آل نیست لذا ما در تصاویر دارای اعوجاج خواهیم بود. دو نوع اعوجاج داریم: اعوجاج شعاعی که حاصل از شکل لنز است و اعوجاج مماسی که حاصل از فرآیند سوار کردن دوربین است. در این تصویر شاهد اعوجاج شعاعی هستیم زیرا خطوط صاف به خطوط منحنی تبدیل شده‌اند و در ضلع پایین و ضلع چپ میز، این انحنا قابل مشاهده است. با تابع cv2.imread() آن را خوانده و با تابع cv2.imshow() آن را نمایش می‌دهیم. برای نمایش بهتر می‌توانیم سایز عکس را به کمک تابع cv2.resize() کمی کوچک کنیم.

```
#Q4_1
path1 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img1.png'
image1 = cv2.imread(path1)
cv2.imshow('image1', image1)
cv2.waitKey(0)
```

۲-۴. با استفاده از تابع findChessboardCorners() می‌توانیم مختصات نقاط گوشه‌ی صفحه‌ی شطرنجی را پیدا کنیم. پارامترهای ورودی این تابع عکس خوانده‌شده، سایز صفحه‌ی شطرنجی (یک واحد کمتر چون نقاط درون را پیدا می‌کند مثلا اینجا ۲۴در۱۷ است) و یک متغیر flag است که می‌تواند صفر یا ترکیبی از آپشن‌های موجود باشد. در آخر خروجی بصورت یک زوج از یک متغیر bool و آرایه‌ای از مختصات گوشه‌ها است. اگر الگوی صفحه را پیدا کند متغیر خروجی True و در غیر این صورت False خواهد شد. در انتها آرایه‌ی corners را چاپ کردیم و مختصات گوشه‌ها نمایش داده شد. (در تصویر زیر بخشی از آن افتاده است.)

```
#Q4_2
flags = cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK +
cv2.CALIB_CB_NORMALIZE_IMAGE
found, corners = cv2.findChessboardCorners(image1, (24, 17))
print(corners)
```

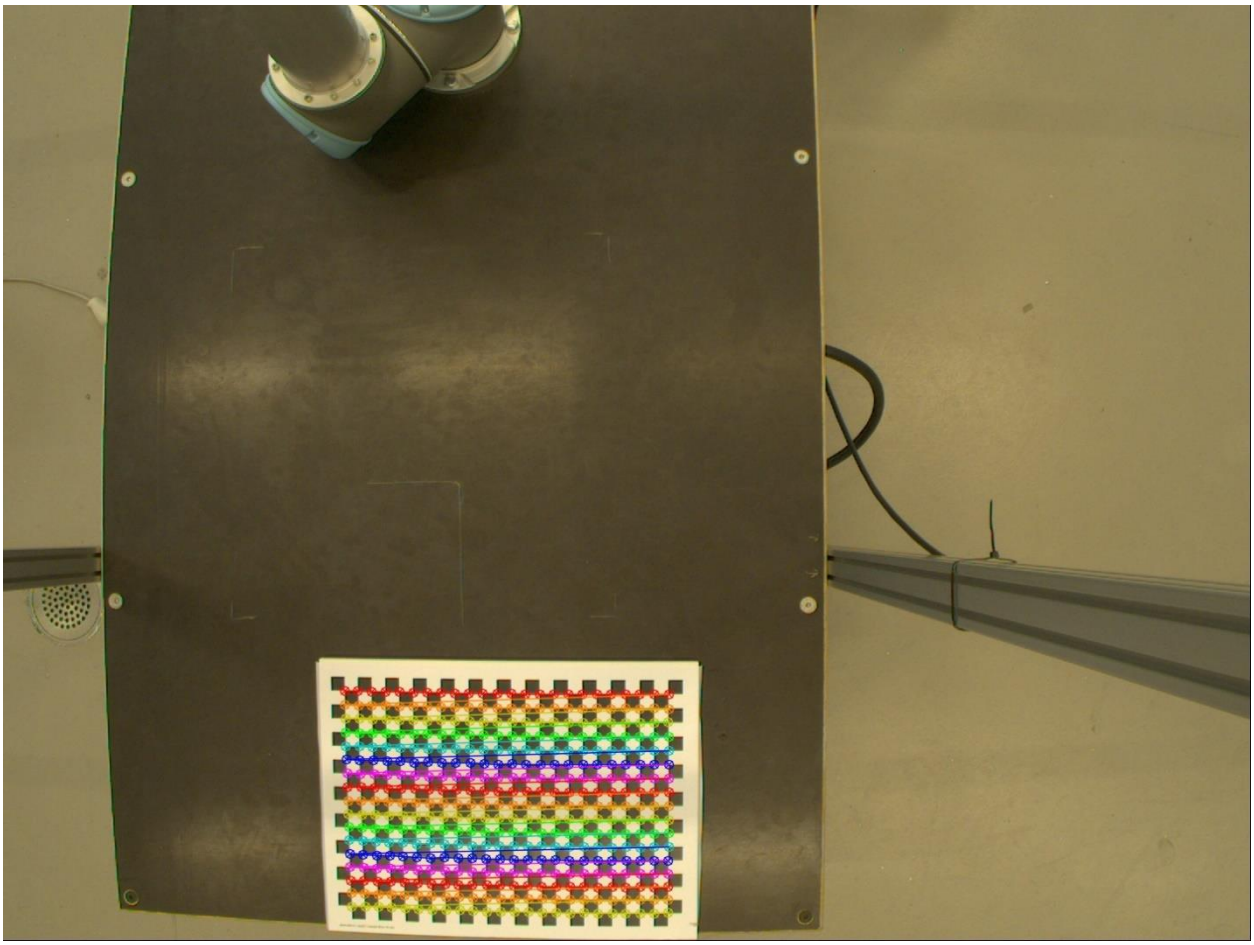
```
image1 = cv2.imread(path1)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[[ 558.3859  1045.7307 ]]
[[ 574.1423  1045.7172 ]]
[[ 590.0136  1046.0675 ]]
[[ 605.9189  1046.3584 ]]
[[ 622.1305  1047.0583 ]]
[[ 638.08563 1046.7454 ]]
[[ 654.3151  1047.5559 ]]
[[ 670.2955  1047.3766 ]]
[[ 686.36523 1047.8539 ]]
[[ 702.5295  1047.5913 ]]
[[ 718.40186 1047.9703 ]]
[[ 734.56525 1047.5602 ]]
```

۴-۳. تابع cv2.cornerSubpix() پنج پارامتر ورودی دارد که به ترتیب خود عکس، آرایه‌ی گوشه‌ها، winSize، zeroZone و criteria هستند. winSize نصف پنجره‌ی سرچ و zeroZone نصف اندازه‌ی dead region و criteria برای جلوگیری از ادامه‌ی الگوریتم در مقدار معینی از تکرار، پس از رسیدن به دقت مشخصی است. سپس با استفاده از تابع cv2.imwrite() آن را ذخیره می‌کنیم. معیار (-1,-1) نشان می‌دهد که هیچ معیاری برای پایان دادن به فرآیند تکراری اصلاح گوشه وجود ندارد.

```
#Q4_3
criteria = (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
gray_image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
corners2 = cv2.cornerSubPix(gray_image1,corners,(5,5),(-1,-1),criteria)
image2 = cv2.drawChessboardCorners(image1, (24,17), corners2, found)
path2 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images'
cv2.imwrite(os.path.join(path2, 'mypic.jpg') , image2)
cv2.imshow('image2', image2)
cv2.waitKey(0)
```



همانطور که در خروجی واضح است تمام گوشه‌ها تشخیص و نمایش داده شده اند.

۴-۴. این تابع سه پارامتر ورودی دارد که به ترتیب objectpoints، imagepoint و imagesize هستند. objectpoints یک آرایه است که هر عنصر آن سه متغیر x,y,z دارد. چون تصویر دو بعدی است، متغیر z همه‌ی آن‌ها صفر است. لذا یک شبکه به x از صفر تا ۲۴ و y از صفر تا ۱۷ را تشکیل می‌دهد. imagesize نیز اندازه‌ی تصویر است. imagepoints هم مختصات گوشه‌هایی است که بدست آورده بودیم. خروجی این تابع پنج پارامتر است که به ترتیب آن‌ها را چاپ نمودیم.

```
objpoints = []
imgpoints = []
objp = np.zeros((24 * 17, 3), np.float32)
objp[:, :2] = np.mgrid[0:24, 0:17].T.reshape(-1, 2)
objpoints.append(objp)
imgpoints.append(corners2)
```

```python
img_size = (image1.shape[1], image1.shape[0])
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
img_size, None, None)
print('ret:', ret)
print('\n****************************************')
print('\nmtx:', mtx)
print('\n****************************************')
print('\ndist:', dist)
print('\n****************************************')
print('\nrvecs:', rvecs)
print('\n****************************************')
print('\ntvecs:', tvecs)
```

```
ret: 0.24172387051056266

****************************************

mtx: [[1.00310884e+03 0.00000000e+00 7.85629641e+02]
 [0.00000000e+00 1.01266495e+03 5.26396913e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

****************************************

dist: [[-0.20492206  0.19492852 -0.00102296 -0.01531432 -0.17725086]]

****************************************

rvecs: (array([[-0.02401163],
       [ 0.03567085],
       [-0.0089398 ]]),)

****************************************

tvecs: (array([[-24.00101488],
       [ 16.20400062],
       [ 60.29082178]]),)
```

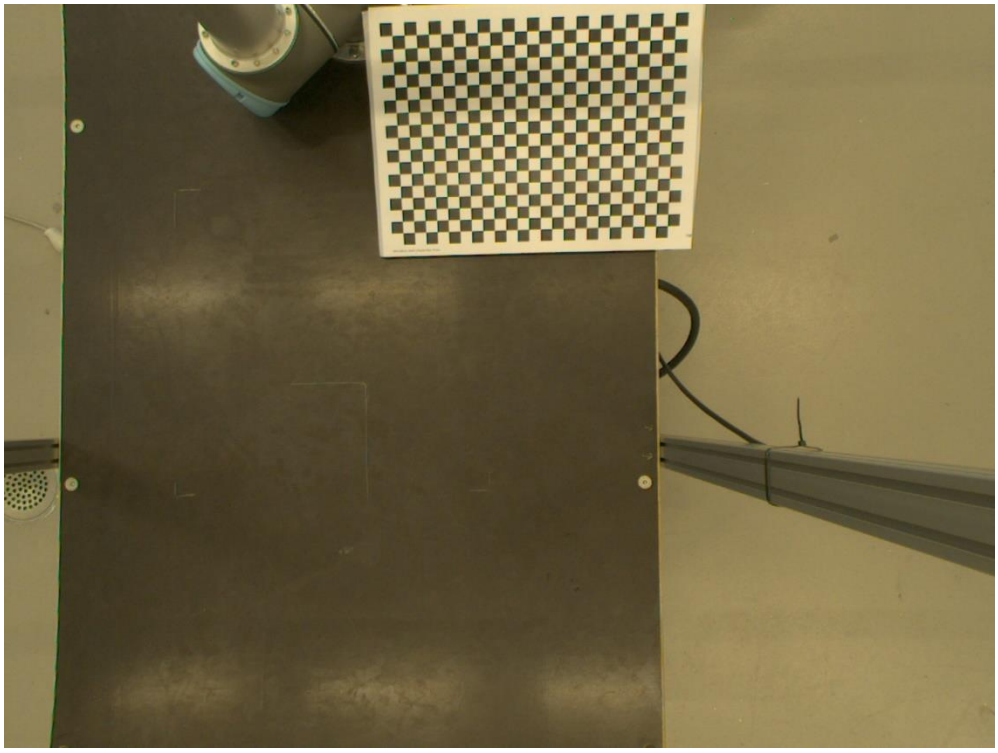
۴-۵) از پارامترهای خروجی بخش قبل، پارامتر dist مربوط به همان ضرایب distortion است.
در واقع برداری از ضرایب است که به ترتیب k1,k2,p1,p2,k3 نمایش داده شده است.

```
print('\nk1:', dist[0][0])
print('\nk2:', dist[0][1])
print('\np1:', dist[0][2])
print('\np2:', dist[0][3])
print('\nk3:', dist[0][4])
```

```
k1: -0.20492205680044595

k2: 0.19492851805357436

p1: -0.0010229644546402657

p2: -0.015314324474244O3

k3: -0.17725086481241956
```

۶-۴) ابتدا ابعاد تصویر را بدست آورده و سپس با استفاده از cv2.getOptimalNewCameraMatrix() یک ماتریس جدید برای دوربین ست می‌کنیم. سپس تصویر گفته شده در صورت سوال را خوانده و با استفاده از تابع cv2.undistort() اعوجاج آن را رفع می‌کنیم. در ورودی این تابع از خود عکس، پارامترهای distortion که از بخش‌های قبلی بدست آورده بودیم و ماتریس جدید دوربین استفاده می‌شود. سپس تصویر جدید را ذخیره می‌کنیم.

```
X, y = image1.shape[:2]
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx,dist,(x,y),0,(x,y))
path3 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img5.png'
image5 = cv2.imread(path3)
dst = cv2.undistort(image5, mtx, dist, None, newcameramtx)
cv2.imwrite(os.path.join(path2, 'undistor.jpg') , dst)
```



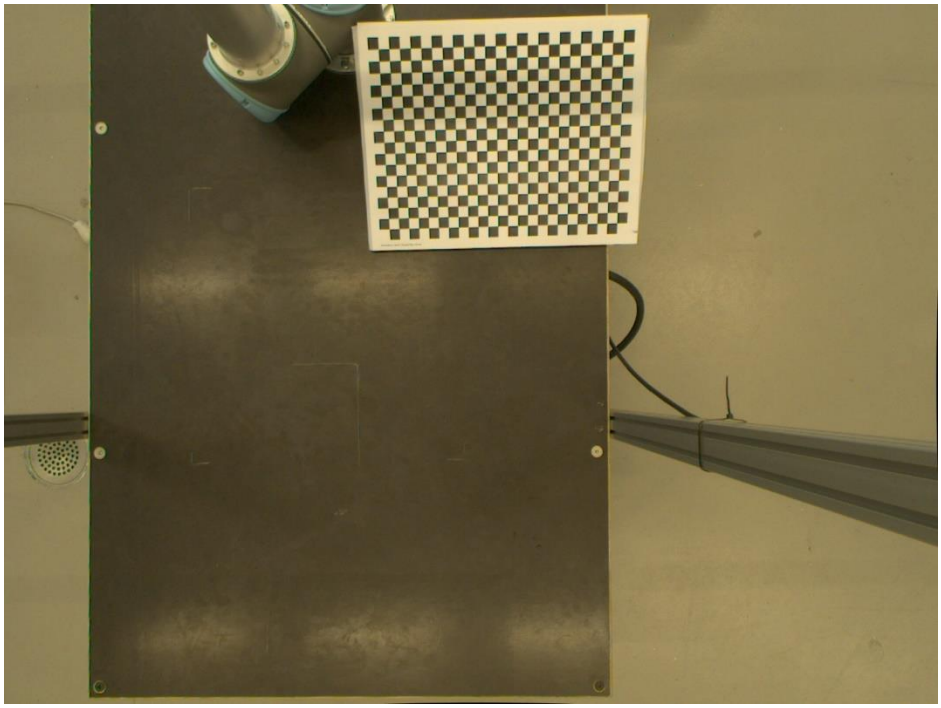
همانطور که ملاحظه می‌فرمایید اعوجاج‌های منحنی تصویر از بین رفته اند.

۴-۷) یک تابع کلی برای رفع اعوجاج‌های تصویر نوشته و پارامترهای کالیبریشن را برای عکس‌های اول تا چهارم بدست می‌آوریم. سپس اعوجاج عکس پنجم را با آن پارامترها حذف می‌کنیم.

```
#Q4_7
objpoints = []
imgpoints = []
img_size = []
def undistort(p, nx, ny):
    img = cv2.imread(p)
    objp = np.zeros((nx * ny, 3), np.float32)
    objp[:, :2] = np.mgrid[0:nx, 0:ny].T.reshape(-1, 2)
    found, corners = cv2.findChessboardCorners(img, (nx, ny))
    if found:
        objpoints.append(objp)
        imgpoints.append(corners)
        img_size.append((img.shape[1], img.shape[0]))

path_img1 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img1.png'
path_img2 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img2.png'
path_img3 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img3.png'
path_img4 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img4.png'
```

```
undistort(path_img1, 24, 17)
undistort(path_img2, 17, 24)
undistort(path_img3, 24, 17)
undistort(path_img4, 24, 17)

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
img_size[0], None, None)
newcameramtx, roi =
cv2.getOptimalNewCameraMatrix(mtx,dist,img_size[0],0,img_size[0])
path_img5 = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images\img5.png'
path_images = r'D:\iust stuff\4011\FCV\homeworks\HW[2]\images'
image5 = cv2.imread(path_img5)
dst = cv2.undistort(image5, mtx, dist, None, newcameramtx)
cv2.imwrite(os.path.join(path_images, 'undistort[1-4].jpg') , dst)
```

تصویر خروجی:



واضح است که خطوط صاف کاملا صاف شده‌اند و نسبت به رفع اعوجاج قبلی به دقت بهتری رسیدیم.