

به نام خدا



درس مبانی بینایی کامپیوتر

---

## تمرین سری دوم

---

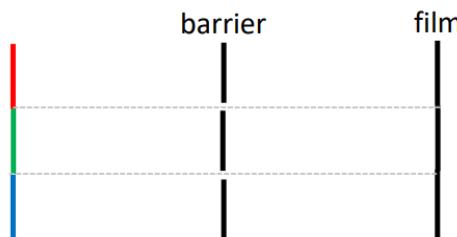
مدرس درس:  
جناب آقای دکتر محمدی

تهیه شده توسط:  
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۰۷/۲۰

## سوال ۱:

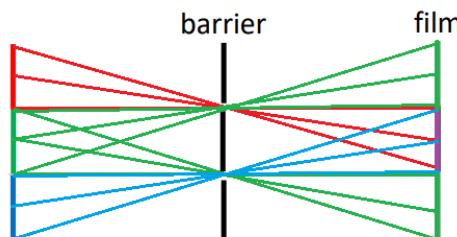
مدل دوربین pinhole ساده‌ترین دستگاه تصویر برداری است. کم و زیاد شدن دریچه در این دوربین چه اثری دارد؟ همانطور که در تصویر قابل مشاهده است یک تغییر کوچکی در این دوربین داده شده و به جای یک دریچه از دو دریچه استفاده شده است. با توجه به اینکه barrier دقیقاً در وسط film و جسم قرار گرفته است، رنگ‌های موجود در شی، تصویر ثبت شده با دوربین را بدست آورید (سنسورهای تصویر برداری از نوع RGB می‌باشند). (۲۰ نمره)



## پاسخ ۱:

زیاد شدن سایز دریچه، باعث اثرگذاری نور منعکس شده در بخش بیشتری از تصویر شده و در نتیجه تصویر تار می‌شود. همچنین کم شدن سایز دریچه، باعث کاهش تار شدن تصویر می‌شود اما مقدار نور ورودی را نیز کم می‌کند و باعث پراکندگی نور می‌شود.

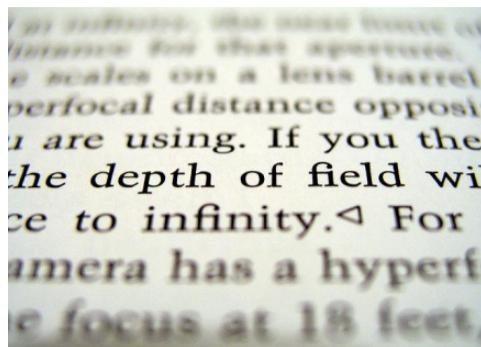
در تصویر موجود، رنگ‌های قرمز، سبز و آبی به ترتیب از بالا به پایین قرار گرفته‌اند. در بخش اول فیلم، رنگ سبز دیده می‌شود. (چون نور منعکس شده از بخش قرمز و آبی، خارج این قسمت قرار می‌گیرد) در بخش دوم، نور قرمز و آبی قرار می‌گیرند که از ترکیب آن‌ها، ارغوانی حاصل می‌شود. در بخش سوم نیز مانند بخش اول، رنگ سبز دیده می‌شود. (چرا که نور منعکس شده از بخش قرمز و آبی، خارج این قسمت می‌افتد). بنابراین، تصویر حاصل، به ترتیب از بالا به پایین سبز، ارغوانی و سبز می‌شود.



شکل حاصل

## سوال ۲:

تصویر زیر با استفاده از یک دوربین ثبت شده است، نوع دوربین استفاده شده برای ثبت تصویر را همراه با ذکر دلیل مشخص کنید (لنزدار یا pinhole). تار بودن یا نبودن قسمت‌های مختلف متن را تفسیر کنید. برای بهبود تصویر چه تغییری می‌توان در روش تصویربرداری ایجاد کرد؟ (۲۰ نمره)



## پاسخ ۲:

با توجه به تصویر، دوربین استقاده شده، لنزدار می‌باشد. چراکه در تصویربرداری با دوربین Pinhole کیفیت تصویر در تمام قسمت‌ها یکسان می‌باشد، یعنی اگر تصویر در یک نقطه تار باشد، در سایر نواحی نیز تار می‌باشد. اما در تصویربرداری با دوربین لنزدار، تصویر در نواحی که در DOF قرار دارند، با وضوح بیشتری نسبت به سایر نواحی دیده می‌شوند. بنابراین با توجه به اینکه تصویر در نواحی میانی واضح‌تر می‌باشد و کیفیت در تمامی قسمت‌ها یکسان نیست، دوربین ما، لنزدار بوده است.

با توجه به وضوح بیشتر تصویر در قسمت میانی، متوجه می‌شویم میدان دید در قسمت میانی واقع است. یعنی به عبارتی فاصله دوربین تا تصویر در بخش میانی، به گونه‌ای است که باعث می‌شود اشیا در این محدوده، واضح‌تر دیده شوند. (مفهوم همان DOF) در قسمت بالایی، به علت فاصله بیشتر دوربین تا آن نواحی، و در قسمت پایینی به علت فاصله کمتر دوربین تا تصویر، باعث می‌شود تصویر در آن نواحی تار دیده شوند. چراکه خارج از محدوده عمق میدان (DOF) قرار گرفته‌اند. برای بهبود این تصویر، می‌توان هم از لنز و هم از دریچه استفاده کنیم تا بتوانیم میدان دید را کنترل کنیم.

## سوال ۳:

می‌خواهیم با استفاده از یک دوربین لنزدار که فاصله کانونی لنز دوربین برابر با  $10\text{ cm}$  می‌باشد، تصویر یک شی را که در فاصله  $40\text{ cm}$  از film می‌باشد ثبت کنیم. فاصله لنز با نیز برابر با  $10\text{ cm}$  است. تصویر ثبت شده به چه شکلی خواهد بود. با فرض ثابت بودن فاصله دوربین با شی، برای ثبت تصویر با کیفیت‌تر لنز دوربین را چگونه باید تنظیم کرد؟ (۲۰ نمره)

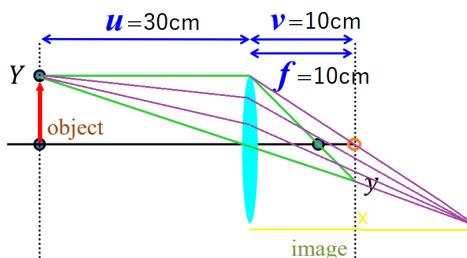
### پاسخ ۳:

با توجه به اطلاعات داده شده داریم:

۱.  $f$  (فاصله کانونی):  $10\text{ cm}$
۲.  $u+v$  (فاصله شی تا فیلم دوربین):  $40\text{ cm}$
۳.  $v$  (فاصله لنز تا فیلم):  $10\text{ cm}$
۴.  $u$  (فاصله شی تا لنز):  $30\text{ cm}$
- از تفریق موردن ۲ و ۳، مقدار  $u$  حاصل می‌شود.
۵. طبق فرمول  $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$  داریم:

$$\frac{1}{10} \neq \frac{1}{30} + \frac{1}{10}$$

از مجموعه معادلات بالا، نتیجه می‌گیریم  $f = v = 10\text{ cm}$  و  $u = 30\text{ cm}$ ، روی هم می‌افتند و در نتیجه، تصویر تار خواهد بود. چرا که برای اینکه تصویر واضح باشد، باید معادله ۵، برقرار باشد. اما در اینجا برقرار نیست. در فاصله به دست آمده برای  $v$  نیز، اشعه‌های نوری همگرا نیستند و در فاصله  $x$  همگرا می‌شوند. با توجه به اینکه اشعه‌های نقطه بالایی شی در چند نواحی از فیلم قرار می‌گیرند، نتیجه می‌گیریم تصویر تار می‌شود. مطابق شکل زیر داریم:



شکل حاصل

با فرض ثابت بودن فاصله دوربین تا شی، یعنی  $v = 10\text{ cm}$  ثابت و برابر  $u = 30\text{ cm}$  سانتی‌متر می‌باشد. با توجه به اینکه فاصله کانونی ( $f$ ) نیز ثابت می‌باشد، بنابراین  $u+v$  باید به گونه‌ای تغییر کنند که رابطه زیر برقرار شود:

با توجه به اینکه  $u+v = 40$ ، مقدار  $u = 20\text{ cm}$  و  $v = 20\text{ cm}$  می‌شود. بنابراین برای ثبت تصویر با کیفیت، باید فاصله لنز تا شی را  $20\text{ cm}$ ، و فاصله لنز تا فیلم را نیز  $20\text{ cm}$  در نظر گرفت.

## سوال ۴:

در این بخش می‌خواهیم پارامترهای مربوط به اعوجاج‌های دوربین را با استفاده از تصاویری که در اختیار داریم تخمین بزنیم. بخش‌های زیر را با توجه به توضیحات پیاده سازی کنید. (۴۰ نمره)

۱. تصاویر داخل پوشه images قرار دارند. تصویر img1.png را به کمک OpenCV خوانده و در مورد اعوجاج تصویر و دلایل آن توضیح دهید.

۲. در تصویر img1.png با استفاده از تابع cv2.findChessboardCorners() الگوی شطرنج را پیدا کرده و نقاط گوشه را استخراج کنید.

۳. برای افزایش دقیق نقاط استخراج شده از تابع cv2.cornerSubPix() استفاده نمایید و نقاط بدست آمده را روی صفحه شطرنجی نمایش دهید. برای این کار از تابع drawChessboardCorners()

۴. با استفاده از نقاط بدست آمده و در دست داشتن مختصات صفحه شطرنجی به کمک تابع cv2.calibrateCamera() پارامترهای مربوط را بدست آورید.

۵. گزارش را k<sub>۱</sub>, k<sub>۲</sub>, p<sub>۱</sub>, p<sub>۲</sub>، پارامترهای کنید.

۶. با استفاده از پارامترهای بدست آمده، اعوجاج تصویر img5.jpg را حذف کنید.

۷. حال برای بدست آوردن پارامترهای کالیبراسیون دوربین تنها از تصویر img[14].png استفاده نمایید و مراحل قبل را تکرار کنید.

۸. نتیجه حاصل برای تصویر img5.jpg را با نتیجه مرحله قبل مقایسه کنید.

#### پاسخ: ۴.۱

اعوجاج در واقع فرآیندی است که منجر به تغییر شکل تصویر می‌شود. لنت ایده‌آل لنتی است که اعوجاج در آن وجود ندارد اما در عمل هیچ لنت ایده‌آلی وجود ندارد. به طور کلی دو نوع اعوجاج اصلی داریم: اعوجاج شعاعی و اعوجاج مماسی. اعوجاج شعاعی حاصل از شکل لنت است و اعوجاج مماسی حاصل از فرآیند سوار کردن دوربین است.

برای خواندن تصویر از دستور cv2.imread() استفاده می‌کنیم که دو آرگومان به عنوان ورودی می‌گیرد که آرگومان اول، مسیر قرار گرفتن عکس، و آرگومان دوم، حالت خوانده شدن عکس است که گذاشت آن اختیاری می‌باشد.

```
4  #part1 (reading img1.png by cv2.imread syntax)
5  #cv2.imread(path, state)
6  path = r'E:\University\Term7\FCV\HW2\images\img1.png'
7  img1 = cv2.imread(path)
8  cv2.imshow('img1', img1)
9  cv2.waitKey(0)
```

کد قسمت اول سوال ۴

#### پاسخ: ۴.۲

در این بخش با استفاده از تابع cv2.findChessboardCorners() نقاط گوشه داخلی شطرنج را پیدا می‌کنیم. در قسمت اول ورودی این تابع، تصویر مورد نظر، بخش دوم، تعداد نقاط گوشه داخلی در هر سطر و ستون، در بخش سوم، آرایه خروجی گوشه‌های شناسایی شده، و بخش چهارم که گذاشت آن اختیاری است، اگرها ای عملیات مختلف که می‌تواند صفر یا ترکیبی از آن‌ها باشد. خروجی این تابع نیز نقاط گوشه و یک bool می‌باشد که در صورت پیدا شدن گوشه مقدار آن true و در صورت پیدا نشدن، مقدار آن false می‌باشند.

```
12  #part2 (find corners by cv2.findChessboardCorners syntax)
13  #findchessboardcorners (InputArray image, Size patternSize,OutputArray corners, int flags)
14  retreval, corners= cv2.findChessboardCorners(img1, (24,17), None)
15  print(retreval)
16  print(corners)
```

کد قسمت دوم سوال ۴

|                         |                         |                         |
|-------------------------|-------------------------|-------------------------|
| True                    | [[475.38907 441.49884]] | [[406.82953 450.02258]] |
| [[273.47873 439.29468]] | [[486.69806 441.3926 ]] | [[418.3152 450.28275]]  |
| [[284.16574 439.329 ]]  | [[498.33368 441.49716]] | [[429.5712 450.33136]]  |
| [[294.76782 439.5938 ]] | [[509.65732 441.41736]] | [[440.95212 450.46097]] |
| [[305.75412 439.63797]] | [[521.2642 441.49426]]  | [[452.37082 450.39734]] |
| [[316.84473 439.931 ]]  | [[532.6769 441.4079 ]]  | [[463.8493 450.58142]]  |
| [[328.0067 440.14548]]  | [[273.5425 447.9001 ]]  | [[475.34048 450.49524]] |
| [[339.27518 440.37643]] | [[284.4839 448.41833]]  | [[486.78885 450.56702]] |
| [[350.38315 440.45938]] | [[295.29245 448.4443 ]] | [[498.33026 450.4651 ]] |
| [[361.48135 440.6219 ]] | [[306.02682 448.62503]] | [[509.70987 450.60596]] |
| [[372.63412 440.5569 ]] | [[317.12686 448.80383]] | [[521.2036 450.47412]]  |
| [[383.7458 440.86276]]  | [[328.23083 449.02704]] | [[532.7428 450.5191 ]]  |
| [[395.28906 440.9151 ]] | [[339.3644 449.30426]]  | [[273.823 456.93417]]   |
| [[406.64862 441.1212 ]] | [[350.46887 449.50946]] | [[284.7154 457.20572]]  |
| [[418.12854 441.21402]] | [[361.5977 449.48102]]  | [[295.5785 457.49997]]  |
| [[429.48877 441.26025]] | [[372.69144 449.65823]] | [[306.47464 457.5439 ]] |
| [[440.762 441.36118]]   | [[384.0965 449.7779 ]]  | [[317.4372 457.73004]]  |
| [[452.33908 441.4163 ]] | [[395.51138 450.0432 ]] | [[328.365 457.85867]]   |
| [[463.75085 441.38614]] |                         |                         |

#### بخشی از نتیجه قسمت دوم سوال ۴

#### پاسخ: ۴.۳

در این بخش، ابتدا با استفاده از تابع cv2.cvtColor() تصویر خود را به حالت سیاه و سفید در می‌آوریم، چرا که برای استفاده از تصویر در تابع cv2.cornerSubPix()، تصویر ورودی باید به حالت سیاه و سفید باشد. سپس دقیق مورد نظر برای توقف فرآیند cv2.cornerSubPix را تعیین می‌کنیم که در اینجا دقیق مورد نظر را ۰.۰۰۱ یا پس از ۱۰۰ iteration قرار داده‌ایم. سپس با استفاده از تابع cv2.cornerSubPix()، مختصات cornerهای جدید را به دست می‌آوریم. ورودی‌های این تابع به ترتیب، تصویر ورودی در حالت سیاه و سفید، آرایه‌ای از cornerهای مورد نظر، نیمی از طول پنجره جستجو، نیمی از منطقه مرده در وسط منطقه جستجو که در اینجا ما مقدار (-1,-1) را قرار داده‌ایم و به این معنی است که همه‌چیز منطقه‌ای وجود ندارد و شاخصی برای ادامه الگوریتم

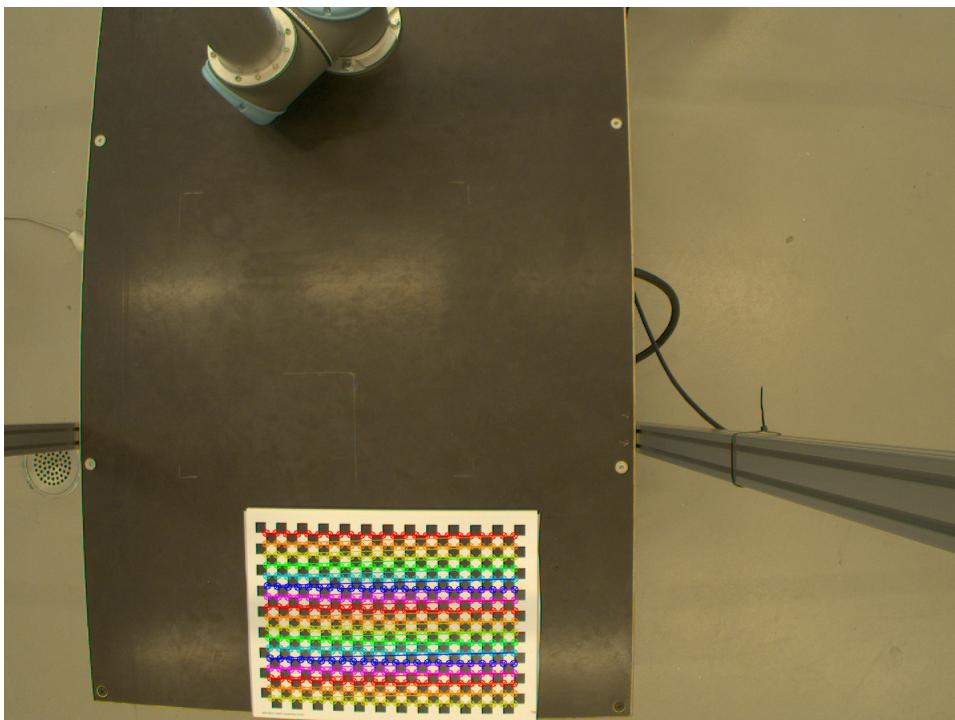
می باشد که درباره آن توضیح دادیم. خروجی این تابع نیز، مختصات گوشه هایی است که توسط تابع پیدا شده است. سپس چک کرده و در صورت پیدا شدن نقاط گوشه، آن ها را با استفاده از تابع cv2.drawChessboardCorners() نمایش می دهیم. ورودی های این تابع نیز، به ترتیب تصویر ورودی، تعداد نقاط داخلی گوشه های شطرنج در هر سطر و ستون، corner هایی که از خروجی تابع قبلی پیدا کردیم، و مقدار retnerval تابع cv2.findChessboardCorners() می باشد

```

18 #part3 (increase the accuracy of found corners by cv2.cornerSubPix)
19 #cornerSubPix (InputArray image, InputOutputArray corners, Size winSize, Size zeroZone, TermCriteria criteria)
20 #drawChessboardCorners (InputOutputArray image, Size patternSize, InputArray corners, bool patternWasFound)
21 gray = cv2.cvtColor(img1 ,cv2.COLOR_BGR2GRAY)
22 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
23 corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1), criteria)
24 imgSubpix = cv2.drawChessboardCorners(img1, (24,17), corners2, retval)
25 cv2.imshow('imgSubpix', imgSubpix)
26 cv2.waitKey(0)
27 pathAnswers = r'E:\University\Term7\FCV\HW2\images'
28 cv2.imwrite(os.path.join(pathAnswers, '4.3_Answer.png'), imgSubpix)

```

#### کد قسمت سوم سوال ۴



نتیجه قسمت سوم سوال ۴

#### پاسخ: ۴.۴

در این قسمت از ما خواسته شده تا با استفاده از تابع cv2.calibrateCamera، پارامترهای مربوط را به دست آوریم. ورودی های این تابع به ترتیب عبارتند از: بردار بردار نقاط الگوی کالیبراسیون در

فضای مختصات الگوی کالیبراسیون که شامل سه پارامتر  $x$  و  $y$  و  $z$  می‌باشد که در تصاویر دو بعدی مانند اینجا، مقدار  $z$ ، صفر می‌باشد. ورودی بعدی، بردار بردارهای پیش‌بینی نقاط الگوی کالیبراسیون هست. ورودی بعدی یا imageSize که از نامش پیداست، اندازه تصویر می‌باشد که برای مقداردهی اولیه ماتریس ذاتی دوربین استفاده می‌شود. cameraMatrix نیز یک ماتریس ۳ در ۳ از دوربین می‌باشد. پارامتری بعدی نیز distCoeffs می‌باشد که ضرایب اعوجاج تصویر است. ورودی‌های بعدی نیز در کد مربوط به این بخش کامنت شده‌اند که الیه گذاشت آن‌ها اختیاری می‌باشد. خروجی این تابع نیز camera matrix، ضریب اعوجاج، بردارهای چرخش و بردارهای انتقال می‌باشد.

```

31 #part4 (determine camera calibration parameters by cv2.calibrateCamera)
32 #calibrateCamera (InputArrayOfArrays objectPoints, InputArrayOfArrays imagePoints, Size imageSize,
33 #                           InputOutputArray cameraMatrix, InputOutputArray distCoeffs,
34 #                           OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs, int flags = 0, TermCriteria)
35 objectPoints = []
36 imagePoints = []
37 imagePoints.append(corners2)
38 #numpy.zeros(shape, dtype=float, order='C', *, like=None)
39 objectP = numpy.zeros((24*17,3), numpy.float32)
40 #print(objectP)
41 objectP[:, :2] = numpy.mgrid[0:24,0:17].T.reshape(-1,2)
42 objectPoints.append(objectP)
43 ret, camMat, distortion, rotV, transV = cv2.calibrateCamera(objectPoints, imagePoints, img1.shape[1::-1], None, None)
44 print(['retrval:\n', ret, '\n\ncamera matrix:\n', camMat, '\n\nndistortion coefficiece:\n', distortion
45           , '\n\nrotation vectors:\n', rotV, '\n\ntranslation vectors:\n', transV])

```

کد قسمت چهارم سوال ۴

```

retrval:
0.22975377370739775

camera matrix:
[[1.00520523e+03 0.0000000e+00 7.91264339e+02]
 [0.0000000e+00 1.01544867e+03 5.20980956e+02]
 [0.0000000e+00 0.0000000e+00 1.0000000e+00]]

distortion coefficiece:
[[-0.20625468  0.1989532 -0.00082261 -0.01649139 -0.18016533]]

rotation vectors:
(array([[-0.02364562],
       [ 0.03777226],
       [-0.00948411]]),)

translation vectors:
(array([[-24.33164186],
       [ 16.51999428],
       [ 60.43366947]]),)

```

نتیجه قسمت چهارم سوال ۴

#### پاسخ : ۴.۵

این پارامترها از قسمت distortion coefficience تابع cv2.calibrateCamera() به دست می‌آیند. k1, k2, p1, p2, k3 به ترتیب از اول تا آخر در خانه‌های این آرایه قرار گرفته‌اند.

```
47 #part5 (determine k1, k2, k3, p1, p2)
48 print('distortion coefficience:')
49 print(['k1:', distortion[0][0], '\nk2:', distortion[0][1], '\n\np1:', distortion[0][2],
50       '\n\np2:', distortion[0][3], '\nk3:', distortion[0][4], '\n\n'])
```

کد قسمت پنجم سوال ۴

```
distortion coefficience:
k1: -0.20625468220093468
k2: 0.19895320011233164
p1: -0.0008226067017545929
p2: -0.016491390046288943
k3: -0.18016533311994923
```

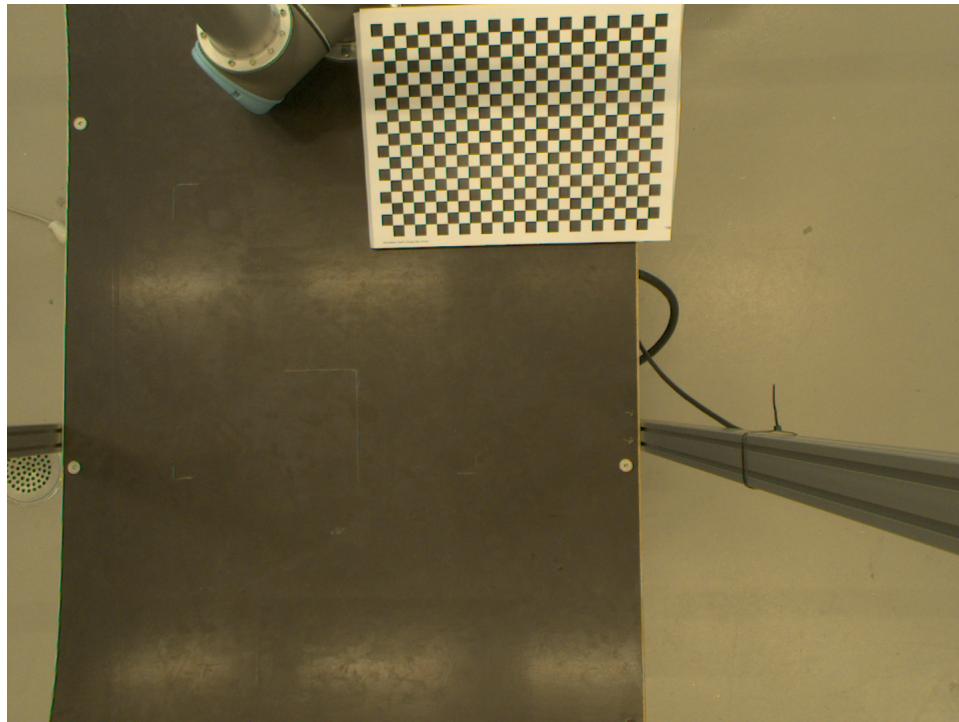
نتیجه قسمت پنجم سوال ۴

#### پاسخ : ۴.۶

در این بخش، ابتدا تصویر ۵ را خوانده و سپس با استفاده از تابع cv2.getOptimalNewCameraMatrix()، ماتریس ذاتی دوربین جدید را براساس پارامترهای مقیاس‌گذاری برمی‌گردانیم. ورودی‌های این تابع، به ترتیب شامل ماتریس دوربین ورودی، برداری از ضرایب اعوجاج، سایز تصویر اصلی، و یک پارامتر مقیاس بندی بین ۰ تا ۱ می‌باشد. سپس با استفاده از تابع cv2.undisort()، اعوجاج را از تصویرمان حذف می‌کنیم. ورودی‌های این تابع نیز شامل تصویر ورودی، ماتریس دوربین ورودی، ضرایب اعوجاج، و ماتریس دوربین جدید می‌باشند.

```
52 #part6 (Undistortion image5 by distortion parameters)
53 img5 = cv2.imread(os.path.join(pathAnswers, 'img5.png'))
54 #cv2.imshow('img5', img5)
55 #cv2.waitKey(0)
56 #getOptimalNewCameraMatrix (cameraMatrix, distCoeffs, imageSize, alpha[, newImgSize[, centerPrincipalPoint]])
57 camMat2, roi = cv2.getOptimalNewCameraMatrix(camMat, distortion, img5.shape[1::-1], 0, img5.shape[1::-1])
58 #undistort(src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]])
59 undisort = cv2.undistort(img5, camMat, distortion, None, camMat2)
60 cv2.imshow('Undistorted image', undisort)
61 cv2.waitKey(0)
62 cv2.imwrite(os.path.join(pathAnswers, '4.6_Answer.png'), undisort)
```

کد قسمت ششم سوال ۴



نتیجه قسمت ششم سوال ۴

پاسخ: ۴.۷

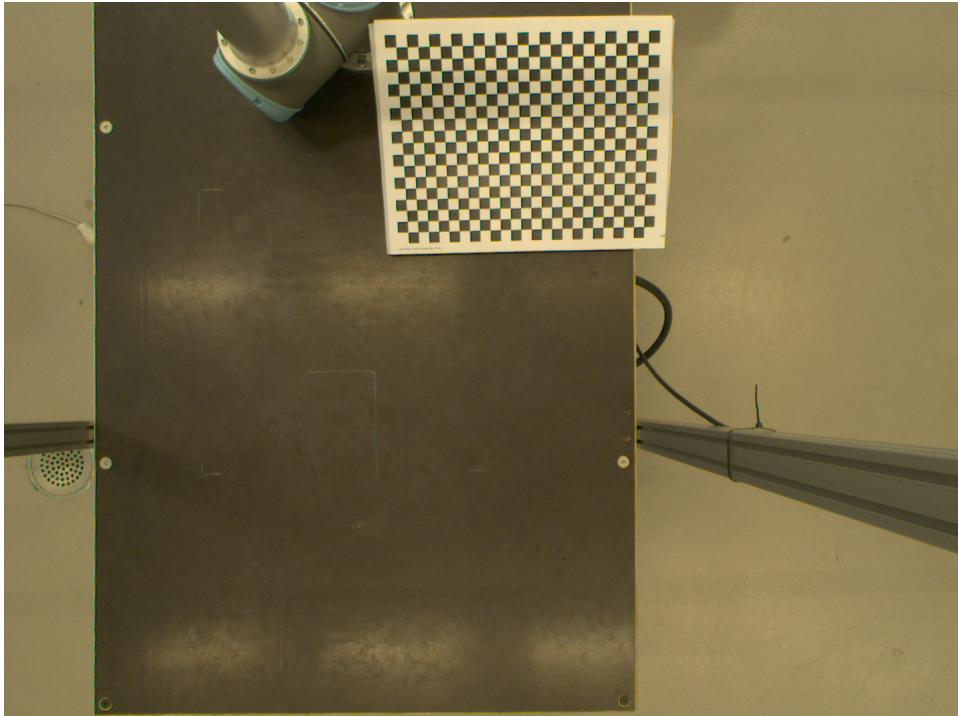
در این بخش همان مراحل قبلی را برای تصاویر ۱ تا ۴ تکرار می‌کنیم و با استفاده از مجموعه مواردی که به دست آوردهیم، بار دیگر پارامترهای کالیبراسیون را به دست آورده و رفع اعوجاج را برای تصویر ۵، یک بار دیگر انجام می‌دهیم.

```

64 #part7 (determin calibration parameters with images[1-4])
65 img1 = cv2.imread(os.path.join(pathAnswers, 'img1.png'))
66 img2 = cv2.imread(os.path.join(pathAnswers, 'img2.png'))
67 img3 = cv2.imread(os.path.join(pathAnswers, 'img3.png'))
68 img4 = cv2.imread(os.path.join(pathAnswers, 'img4.png'))
69 img5 = cv2.imread(os.path.join(pathAnswers, 'img5.png'))
70 images = [img1, img3, img4]
71 objPoints = []
72 imgPoints = []
73 objectP2 = numpy.zeros((17*24,3), numpy.float32)
74 objectP2[:, :2] = numpy.mgrid[0:17, 0:24].T.reshape(-1, 2)
75 for image in images:
76     reterval, corners = cv2.findChessboardCorners(image, (24, 17))
77     if reterval == True:
78         objPoints.append(objectP2)
79         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
80         corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
81         imgPoints.append(corners2)
82     reterval, corners = cv2.findChessboardCorners(img2, (17, 24))
83     if reterval == True:
84         objPoints.append(objectP2)
85         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
86         corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
87         imgPoints.append(corners2)
88 ret2, camMat3, distortion2, rotV2, transV2 = cv2.calibrateCamera(objPoints, imgPoints, img1.shape[1::-1], None, None)
89 camMat4, roi = cv2.getOptimalNewCameraMatrix(camMat3, distortion2, img1.shape[1::-1], 0, img1.shape[1::-1])
90 undistortion2 = cv2.undistort(img5, camMat3, distortion2, None, camMat4)
91 cv2.imshow('Undistorted image', undistortion2)
92 cv2.waitKey(0)
93 cv2.imwrite(os.path.join(pathAnswers, '4.7_Answer.png'), undistortion2)

```

کد قسمت هفتم سوال ۴



نتیجه قسمت هفتم سوال ۴

#### پاسخ: ۴.۸

با مقایسه تصویر حاصل از مرحله ۷ و مرحله ۶، متوجه می‌شویم تصویر مرحله ۷ اعوجاج را به طور بهتری حذف کرده است و دلیل آن، استفاده از نمونه‌های بیشتر برای رفع اعوجاج بوده است.