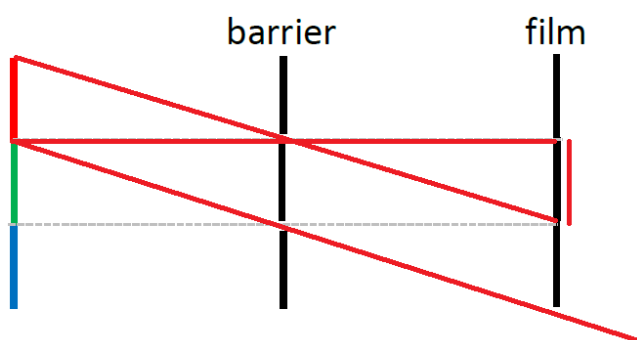


## ۱.الف) مدل دوربین pinhole ساده‌ترین دستگاه تصویر برداری است. کم و زیاد شدن دریچه در این دوربین چه اثری

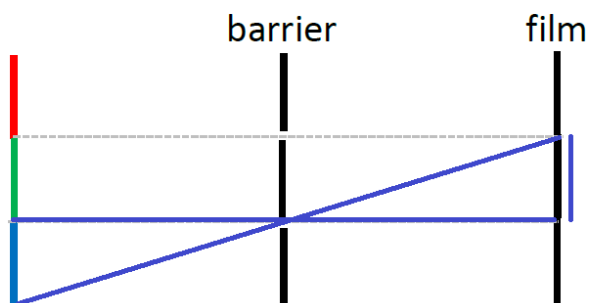
**دارد؟** بزرگ‌تر کردن دریچه باعث تار شدن تصویر ثبت شده می‌شود چون نور بازتاب شده از یک نقطه از تصویر ممکن است به دو نقطه‌ی مختلف از film بتابد و این پدیده باعث می‌شود با ترکیب شدن آن دو نقطه از صحنه در یک نقطه از تصویر ثبت شده تاری تصویر بوجود بیاید. با کوچک کردن دریچه می‌توانیم تاری تصویر را کم کنیم اما چون مقدار نوری که به دوربین وارد می‌شود را کم کرده‌ایم، تاثیر نویز روی آن بیشتر خواهد بود و اگر از حدی بیشتر دریچه را کوچک کنیم بدلیل عبور نور از یک ناحیه بسیار باریک پدیده فیزیکی پراکندگی نور ثبت تصویر را با مشکل مواجه خواهد کرد که باز هم باعث تاری تصویر می‌شود.

## ۱.ب) ... تصویر ثبت شده با دوربین را بدست آورید.

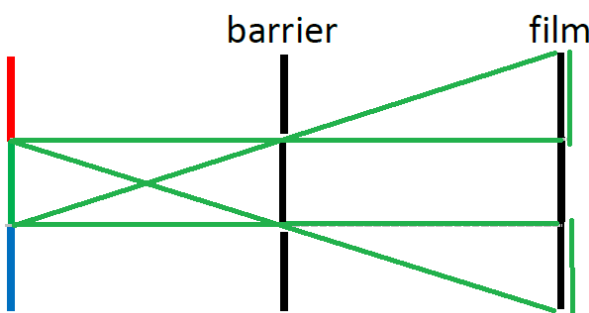
نور بازتاب شده از قسمت قرمز رنگ از دریچه بالایی به قسمت وسطی فیلم می‌رسد. محل قرارگیری قسمت قرمز رنگ بطوری است که از دریچه پایینی نور منعکس شده از آن به فیلم نمی‌رسد.



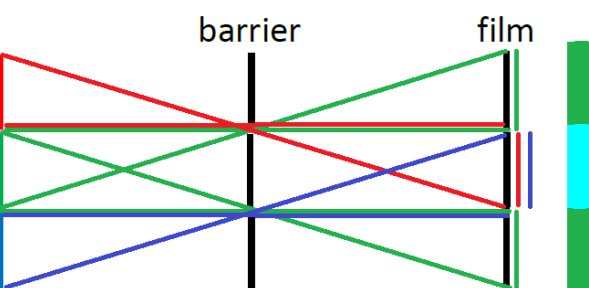
بطور مشابه نور بازتاب شده از قسمت آبی رنگ شی هم فقط از دریچه پایینی و فقط به قسمت وسطی فیلم می‌رسد.



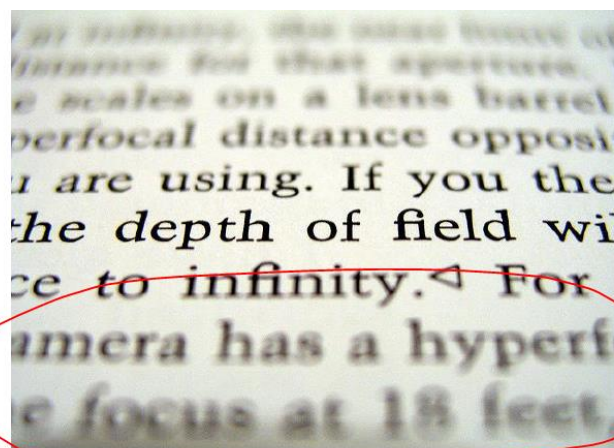
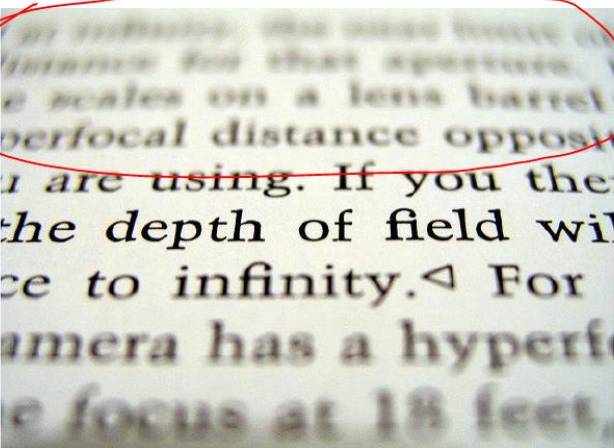
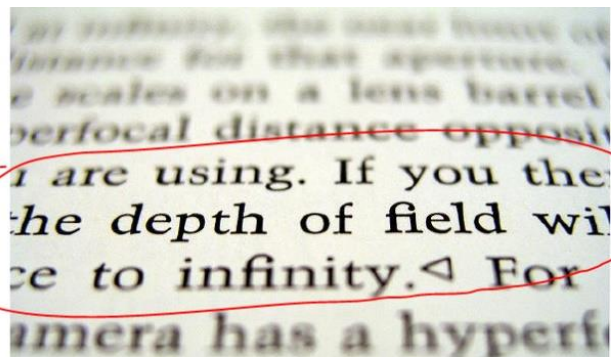
نور بازتاب شده از قسمت سبزرنگ از دریچه بالایی به قسمت بالایی فیلم و از دریچه پایینی به قسمت پایینی فیلم می‌رسد اما به قسمت وسطی فیلم نمی‌رسد.



در مجموع با توجه به تحلیل جداگانه‌ای که در بالا برای هر یک از سه قسمت رنگی شی انجام شد می‌توانیم بگوییم تصویر نهایی احتمالا به این شکل ثبت خواهد شد که قسمت بالا و پایینی تصویر به رنگ سبز خواهد بود و قسمت وسطی چون هم نور طیف قرمز رنگ و هم آبی رنگ تابیده پس احتمالا به رنگ فیروزه‌ای خواهد بود.



۲. دوربین لنزدار استفاده شده است. چون مشکل محدود بودن عمق میدان بدلیل همگرانشدن و متمرکز نشدن انعکاس نوری است که از شی مورد نظر به لنز دوربین لنزدار می تابد و هر شی ای که نزدیک به فاصله  $u$  با شرط معادله لنز نازک ( $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$ ) قرار نداشته باشد بطور متمرکز و صاف در تصویر ثبت نمی شود چون تصویر تابیده شده به نقاط مختلف لنز، در نقطه ای غیر از روی فیلم عکاسی همگرا می شود که مطلوب نیست.



قسمتی از متن که تقریباً در وسط قرار دارد و بصورت نسبتاً واضح ثبت شده است احتمالاً در فاصله ی تقریباً  $u$  از لنز دوربین قرار گرفته بطوری که معادله ( $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$ ) برقرار باشد (در این معادله  $v$  فاصله لنز دوربین از صفحه فیلم عکاسی و  $f$  فاصله کانونی لنز است)

همچنین قسمت هایی که در فاصله بیشتری قرار گرفته اند یعنی  $u$  آن ها بیشتر است پس برای اینکه معادله لنز برای آن ها برقرار بشود باید  $v$  کمتری داشته باشد. پس تار شدن تصویر آن قسمت از متن که در بالای تصویر قرار گرفته و فاصله بیشتری با لنز دارد به علت این است که تصویر تابیده شده در نقطه ای بین لنز و فیلم همگرا می شود، نه روی فیلم عکاسی و تا به صفحه فیلم برسد واگرا می شود که علت تار شدن آن است.

بطور مشابه با توضیحات بالا، قسمتی از تصویر که در پایین تصویر قرار گرفته و مربوط به متنی است که نزدیک تر به لنز قرار گرفته چون  $u$  کمی دارد در فاصله دورتری همگرا می شود و وقتی نورهای تابیده شده از آن به صفحه فیلم عکاسی می رسد هنوز همگرا نشده و به همین دلیل کمی تار ثبت می شود.

برای بهبود کیفیت تصویر نیاز داریم کاری کنیم عمق میدان افزایش یابد. به این منظور از ترکیب لنز و دریچه استفاده می کنیم تا قسمت هایی از تصویر و اشیایی که فاصله شان با لنز  $u$  نیست کمتر تار شوند. نحوه عملکرد این راه حل هم به این صورت است که وقتی نور تابیده از یک شی دور یا نزدیک از دریچه عبور کند و سپس به لنز دوربین برسد، طبیعتاً نور عبور کرده از دریچه قسمت های اضافی کمتری دارد و به همین علت تعداد نورهای تابیده شده از شی که واگرایی آن ها بخواهد روی پیکسل های مختلف تاثیر نامطلوب بگذارد بسیار کاهش می یابد و شی در تصویر ثبت شده واضح تر از حالت بدون دریچه ثبت خواهد شد. همچنین با توجه به حالت های مختلف عکاسی و شرایط نوری متفاوت، اندازه دریچه استفاده شده در کنار لنز هم می تواند برای بدست آمدن عمق میدان مطلوب تاثیرگذار باشد.

۳. معادله لنز نازک:  $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$

معادله لنز برای شرایط گفته شده ( $f=10, u=40, v=10$ ) برقرار نیست، پس احتمالا تصویر ثبت شده تا حدی تار خواهد بود.  $\frac{1}{10} \neq \frac{1}{40} + \frac{1}{10}$

از آن جایی که احتمالا تعویض لنز برای تغییر دادن فاصله کانونی هزینه بر خواهد بود پس روش منطقی و در دسترسی تغییر فاصله لنز و film است بطوری که معادله بالا برقرار شود:

$$\frac{1}{10} \approx \frac{1}{40} + \frac{1}{v} \Rightarrow \frac{3}{40} \approx \frac{1}{v} \Rightarrow v \approx 13.33cm$$

در نتیجه برای بهبود کیفیت تصویر ثبت شده از شی می توانیم حدود 3.33 سانتی متر فاصله لنز تا film را افزایش دهیم.

۱.۴. باز کردن تصویر:

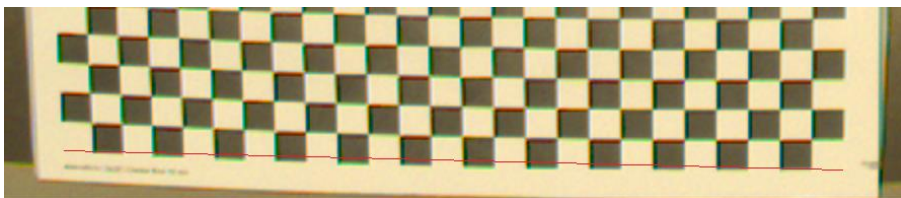
```
In [6]: import cv2
        image = cv2.imread("./images/img1.png", 1)
        image.shape

Out[6]: (1080, 1440, 3)
```

**در مورد اعوجاج تصویر و دلایل آن توضیح دهید.** اعوجاج یا Distortion در تصویر به این معناست که تصویر

ثبت شده از یک شی بطور مصنوعی بزرگ باشد یا خطوط شی با انحنا و بصورت کروی ثبت شوند. یکی از دلایل اعوجاج نزدیکی بیش از حد لنز دوربین به شی است که باعث می شود نقطه ای از تصویر بسیار نزدیک به لنز و نقطه ای دیگر نسبت به آن فاصله،

بسیار دورتر باشد و ایجاد اعوجاج کند. به این نوع از اعوجاج، اعوجاج چشم انداز می گویند که دلیل اصلی آن فاصله و موقعیت مکانی لنز و شی است.



تقریبا هیچ لنز ساخته شده ای ایده آل نیست. در نوع دیگری از اعوجاج که به اعوجاج لنز معروف است، دلیل بوجود آمدن اعوجاج، خطا در ساخت لنز است. این نوع از اعوجاج دو نوع اصلی دارد:

- اعوجاج شعاعی: حاصل از شکل لنز است و معمولا در دوربین های ارزان قیمت بسیار محسوس است. در این نوع از اعوجاج در مرکز تصویر اعوجاج نداریم و با حرکت به سمت گوشه ها و مرزهای تصویر می رویم اعوجاج بیشتر و بیشتر می شود و خطوط منحنی تر دیده می شوند.
- اعوجاج مماسی: حاصل از فرآیند سوار کردن دوربین است که صفحه فیلم دوربین بصورت کاملا موازی با لنز سوار نشود و تا حدی زاویه دار باشد که هر چه بیشتر زاویه دار باشد طبیعتا مقدار اعوجاج مماسی افزایش می یابد.

## ۲.۴

ابتدا تصویر را به حالت grayscale (تک‌کاناله) تغییر دادم و سپس تصویر سیاه و سفید را در ورودی به `findChessboardCorners` دادم. این تابع ابعاد صفحه شطرنجی را هم در ورودی می‌گیرد. که این ابعاد در واقع ابعاد مربع‌های درونی صفحه شطرنجی است و یکی کمتر از ابعاد اصلی صفحه است. برای مثال در اینجا ابعاد صفحه شطرنجی که در تصویر

```
In [20]: gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ret, corners = cv2.findChessboardCorners(gray, (24, 17), None)

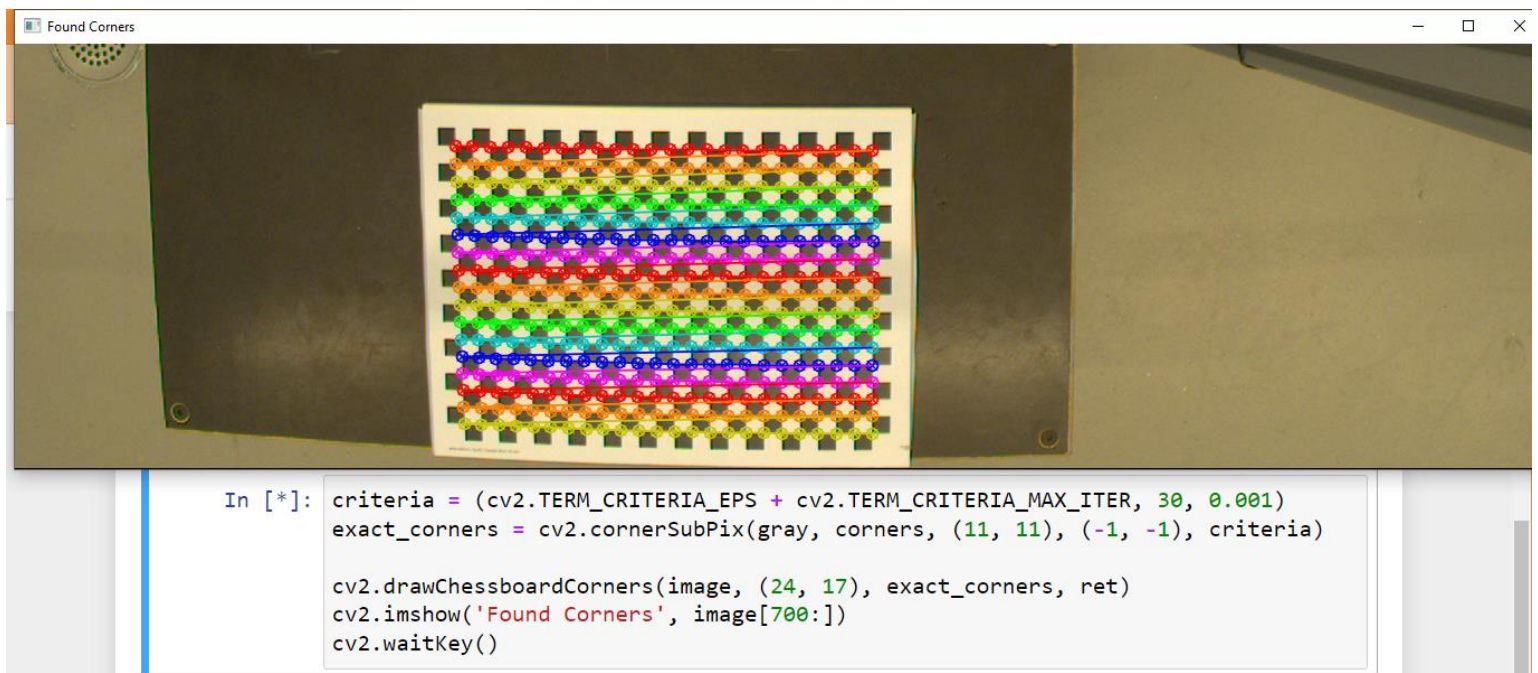
In [25]: print(ret, corners.shape)
corners[0]

True (408, 1, 2)

Out[25]: array([[394.29108, 791.2645 ]], dtype=float32)
```

داریم ۱۸ در ۲۵ است ولی ۱۷ در ۲۴ را به تابع ورودی دادم. ورودی سوم تابع هم نشان‌دهنده آرایه مقصد برای ذخیره‌سازی مختصات گوشه‌ها است که چون مختصات گوشه‌ها را از طریق خروجی تابع در `corners` ذخیره کردم از آن ورودی تابع استفاده نکردم و `None` دادم. در نهایت `ret` برابر `True` است، به این معنا که صفحه شطرنجی با ابعاد خواسته شده با موفقیت تشخیص داده شده و در یک آرایه‌ی ۴۰۸ تایی (۱۷×۲۴=۴۰۸) مختصات گوشه‌ها (X,Y) تحویل داده شده‌اند و در `corners` ذخیره کردم.

## ۳.۴



تابع `cornerSubPix` به یک `criteria` نیاز دارد تا بفهمد تا کی باید الگوریتم بهبود مختصات گوشه‌ها را ادامه دهد. به این صورت که اگر اجرای الگوریتم بیش از تعدادی مرحله مشخص (`max_iteration`) که در اینجا مشابه مثال داک `opencv` ۳۰ دادم) طول بکشد ادامه ندهد یا اینکه اگر دقت بهبود یافته در یک مرحله از الگوریتم و مرحله‌ی بعدی آن کوچکتر از `epsilon` بهبود یافت (که `epsilon` را مشابه مثال داک `opencv` ۰.۰۰۱ دادم) الگوریتم را متوقف کند و ادامه ندهد. علاوه بر `criteria` تابع `cornerSubPix` دو پارامتر `winSize` و `zeroZone` را هم در ورودی دریافت می‌کند که پارامترهای مورد استفاده در الگوریتم بهبود مختصات گوشه‌ها هستند. `winSize` نصف سایز پنجره جستجو در الگوریتم است و `zeroZone` نصف سایز ناحیه‌ای است که در میانه اجرای الگوریتم آن را “ناحیه مرده” می‌نامد که اگر (-1,-1) داده شود در نظر می‌گیرد ناحیه مرده نداریم. هر دوی این پارامترها را مشابه مثال داک `opencv` وارد کردم. در نهایت بعد از پاس دادن خود تصویر، مختصات گوشه‌های بدست‌آمده از تابع قبلی و پارامترهای گفته شده در بالا، گوشه‌های بدست‌آمده را روی تصویر اصلی رسم کردم و در نهایت با `imshow` از ردیف ۷۰۰ به بعد تصویر را روی صفحه نمایش دادم که نتیجه را در بالا می‌بینید.



```
In [36]: import numpy as np
objp = np.zeros((24*17,3), np.float32)
objp[:,2] = np.mgrid[0:24,0:17].T.reshape(-1,2)
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera([objp], [corners], gray.shape[::-1], None, None)
ret, mtx, dist, rvecs, tvecs
```

```
Out[36]: (0.2297543291840789,
array([[1.00523034e+03, 0.00000000e+00, 7.91267405e+02],
       [0.00000000e+00, 1.01547403e+03, 5.20946654e+02],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]]),
array([[-0.20625612, 0.19893305, -0.00081862, -0.01649156, -0.18013834]]),
(array([[-0.02364514],
        [ 0.03777357],
        [-0.00948477]]),),
(array([[-24.33180779],
        [ 16.52202812],
        [ 60.43513479]]),),)
```

پارامترهای ورودی تابع `calibrateCamera`:

- `ObjectPoints`: مختصات نقاط صفحه شطرنجی در فضای مختصاتی ۳ بعدی. طبیعتاً مختصه‌ی Z همواره صفر است و X از ۰ تا ۲۳ و Y از ۰ تا ۱۶ شمرده می‌شوند و در یک آرایه `objectPoint` ریخته می‌شود و در نهایت لیستی از این `objectPoint`ها به ازای هر تصویر گرفته شده برای فرآیند `calibration` به تابع داده می‌شود. (در اینجا چون فقط یک تصویر داریم لیست یک‌عضوی داده شده)
- `imagePoints`: لیستی از مختصات گوشه‌ها به ازای تصویرهای مختلف مورد استفاده در `calibration`
- `imageSize`: سایزهای تصویر، با ترتیب برعکس چیزی که `shape` برمی‌گرداند.
- دو پارامتر دیگر که مربوط به تنظیمات جزئی الگوریتم `calibration` هستند اختیاری هستند و `None` داده شده‌اند.

پارامترهای خروجی تابع `calibrateCamera`:

- `ret`: خروجی اول خطای `calibration` است که در یک `calibration` مطلوب باید عددی بین ۰.۱ تا ۱ پیکسل باشد. خوشبختانه در اینجا به حدود ۰.۲ رسیده‌ایم.
- `mtx`: ماتریس مشخصه ذاتی دوربین
- `dist`: ضرایب اعوجاج (که در قسمت بعدی به این ضرایب نیاز داریم)
- `rvecs`: بردار `rotation`
- `tvecs`: بردار `translation`

```
In [49]: k1, k2, p1, p2, k3 = dist[0]
print(f"k1: {k1}\nk2: {k2}\np1: {p1}\np2: {p2}\nk3: {k3}")
```

```
k1: -0.2062561242691053
k2: 0.19893304963023853
p1: -0.0008186200504633758
p2: -0.016491562084045258
k3: -0.18013834016531752
```

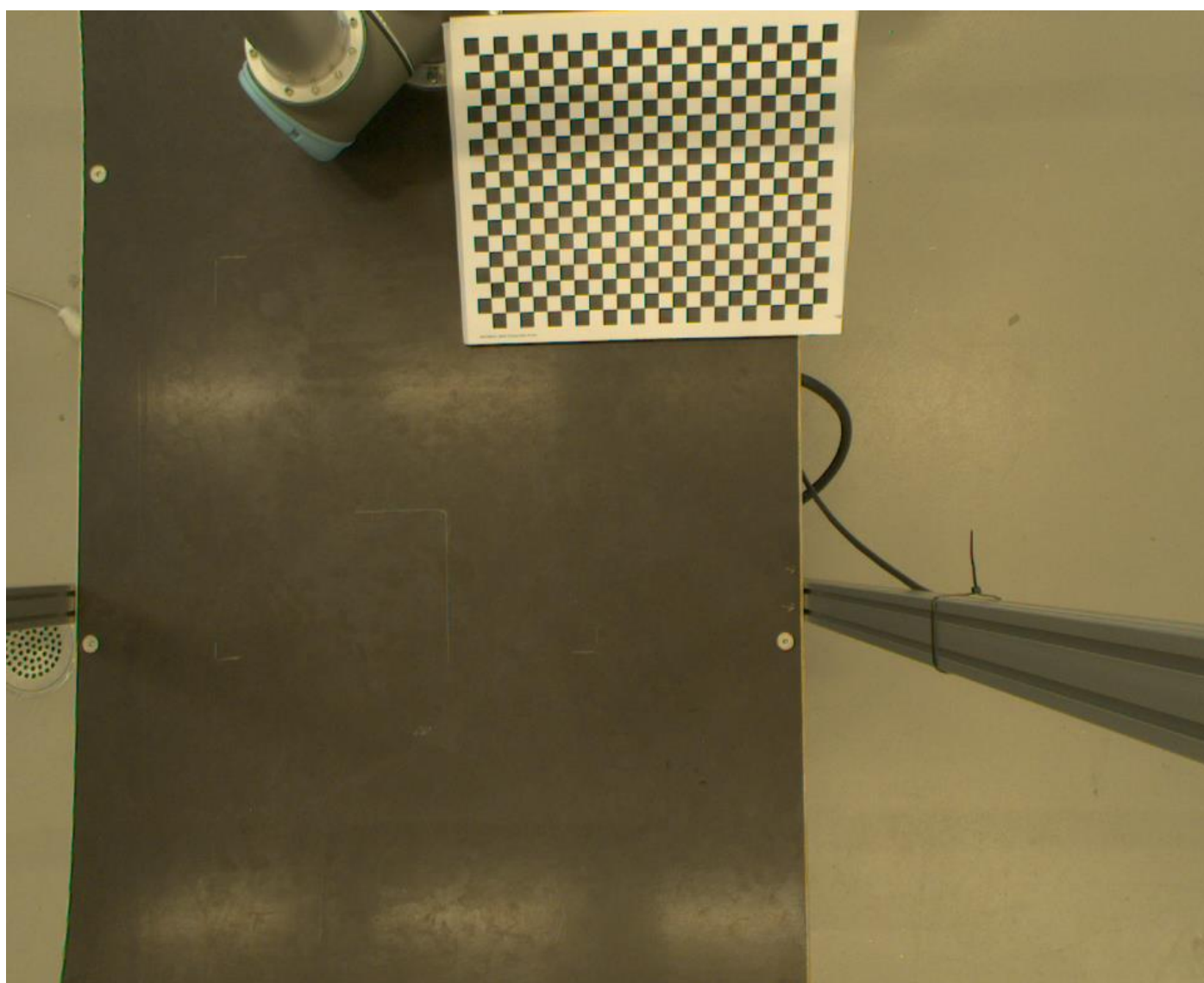
```
In [31]: image5 = cv2.imread("./images/img5.png", 1)
w_h = image5.shape[1::-1]
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, w_h, 1, w_h)

# undistort
dst = cv2.undistort(image5, mtx, dist, None, newcameramtx)

# crop the image
x, y, w, h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibresult1.png', dst)
```

Out[31]: True

تصویر ۵ را باز کردم و ابعاد آن را بدست آوردم و وارونه کردم تا به تابع `getOptimalNewCameraMatrix` بدهم. این تابع با توجه به ماتریس مشخصه دوربین و پارامترهای اعوجاج بدست آمده ماتریس مشخصه جدیدی برای دوربین بدست می‌آورد و ابعاد تصویر جدید را هم برای برش (crop) بعد از اصلاح اعوجاج در اختیار می‌گذارد. سپس به ترتیب اعوجاج برطرف شده و تصویر برش داده شده در فایل `calibresult1.png` ذخیره شده است.



تصویر اصلاح شده `img5` با استفاده از پارامترهای کالیبره `img1`

```
In [37]: images = tuple(cv2.imread(f"./images/img{i}.png", 1) for i in range(1, 4 + 1))
grays = tuple(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) for image in images)
sizes = ((24, 17), (17, 24), (24, 17), (24, 17))
corners_list = tuple(cv2.findChessboardCorners(gray, size, None)[1] for gray, size in zip(grays, sizes))

objpoints = []
for size in sizes:
    objp = np.zeros((24*17,3), np.float32)
    objp[:,2] = np.mgrid[0:size[0],0:size[1]].T.reshape(-1,2)
    objpoints.append(objp)

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, corners_list, gray.shape[:::-1], None, None)

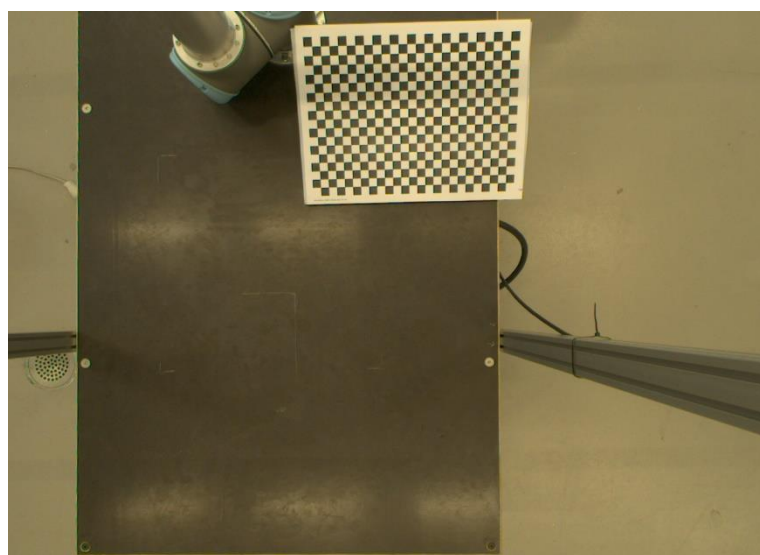
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, w_h, 1, w_h)

# undistort
dst = cv2.undistort(image5, mtx, dist, None, newcameramtx)

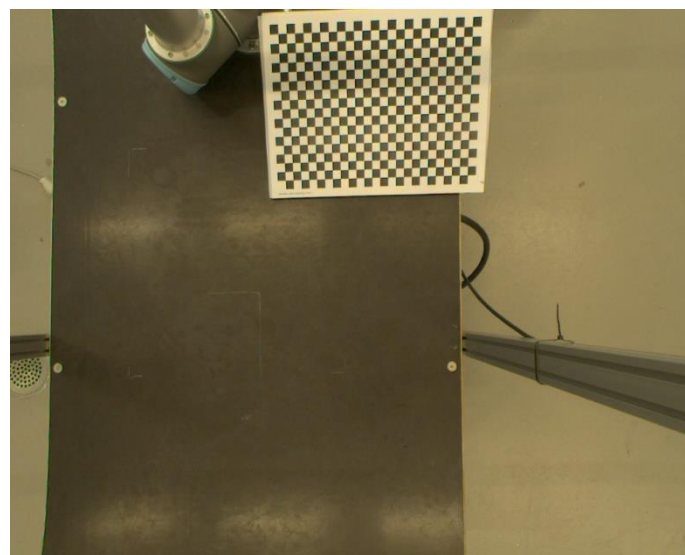
# crop the image
x, y, w, h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibresult2.png', dst)
```

Out[37]: True

تصاویر img1 تا img4 را باز کردم و سیاه‌وسفید شده‌ی آن‌ها را در grays ذخیره کردم. در تصویر img2 برخلاف سه تصویر دیگر، صفحه شطرنج بصورت عمودی قرار گرفته است. پس سائیزهای ذخیره شده برای دیگر imgها ۱۷ در ۲۴ و img2 را ۲۴ در ۱۷ گرفتم. با استفاده از سائیزهای ذخیره شده و تصاویر سیاه‌وسفید شده از تابع findChessboardCorners کمک گرفتم و در corners\_list[i] مختصات گوشه‌های تصویر نام را ذخیره کردم. مشابه قسمت ۴.۴ در اینجا هم بر حسب سائیز صفحات شطرنج object pointها (نقطه‌ها در صفحه مختصاتی) را ایجاد کردم که در مورد img2 چون عمودی قرار گرفته [size[0] و [size[1] طبیعتاً برعکس بقیه‌ی imgها است. در نهایت با لیست آماده شده از objpointها و مختصات گوشه‌ها به ازای هر تصویر، دوربین را کالیبره کردم و پارامترهای مربوط را برای اصلاح اعوجاج img5 پاس دادم.



img5 بعد از اصلاح بر اساس پارامترهای کالیبره img1-img4



img5 بعد از اصلاح بر اساس پارامترهای کالیبره img1

همانطور که از مقایسه دو تصویر آشکار است وقتی که فقط از پارامترهای کالیبره یک تصویر استفاده کردیم لبه‌های میز بصورت منحنی و تصویر دارای اعوجاج است ولی در تصویر سمت چپ که حاصل اصلاح اعوجاج بر اساس پارامترهای کالیبره ۴ تصویر است که صفحه شطرنج در هر یک از آن‌ها در زوایای مختلف و مکان‌های مختلفی قرار دارد نتیجه‌ی نهایی دارای خطوط بسیار صاف‌تر است و اعوجاج بسیار با دقت بیشتری اصلاح شده است. نتیجه این که اگر از یک تصویر برای اصلاح اعوجاج استفاده کنیم شاید کافی نباشد و بهتر است چندین تصویر با موقعیت‌های مختلف صفحه شطرنج را برای این کار استفاده کنیم.

## منابع:

[OpenCV: Camera Calibration](#) :سوال ۲.۴

[OpenCV: Camera Calibration and 3D Reconstruction](#)

[OpenCV: cv::TermCriteria Class Reference](#) :سوال ۳.۴

[OpenCV: Feature Detection](#)

[OpenCV: Camera Calibration](#)

[OpenCV: Camera Calibration and 3D Reconstruction](#) :سوال ۴.۴

[python - Meaning of the retval return value in cv2.CalibrateCamera - Stack Overflow](#)

[OpenCV: Camera Calibration](#) :سوال ۵.۴

[OpenCV: Camera Calibration](#) :سوال ۶.۴

[OpenCV: Camera Calibration](#) :سوال ۷.۴