

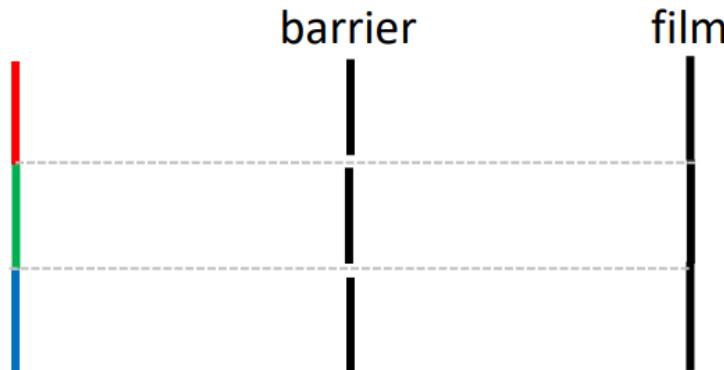
Homework 2

سوال 1

در دوربین pinhole اگر :

دريچه از حدی بزرگتر باشد : چون پرتوهای بيشتری روی صفحه پشت دريچه تاثير ميگذارند تصوير تار می شود.

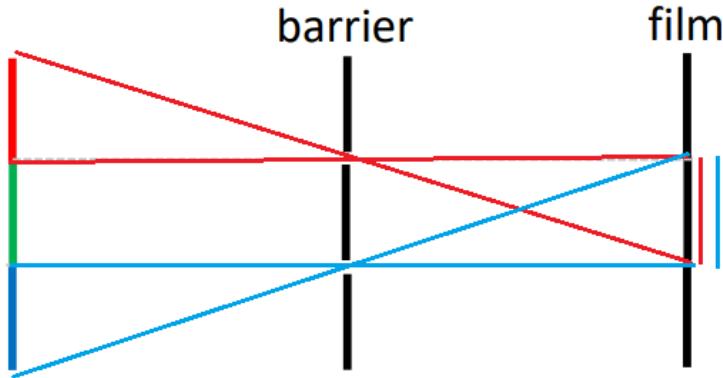
دريچه از حدی کوچکتر باشد : پرتو دچار پراکندگی نوري می شود و باز هم پرتو آنجايی که باید قرار نمي گيرد . همچنین نور کمتری وارد دوربین می شود که در نتیجه هر دوی اينها ، تصوير ناواضح می شود .



در اين شكل ، هر بخش در تصوير (سمت چپ) به بخشی از فيلم که موازي با آن رنگ در رو به روی آن است نمي رسد . در شكل زير پرتو های هر بخش رسم شده است .

رنگ ها به صورت زير هستند :

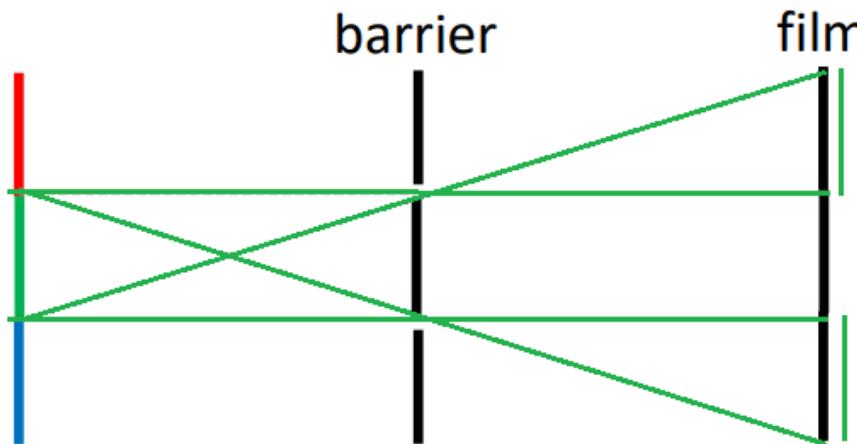
بخش وسطی فيلم :



ترکیبی از آبی و قرمز ← بنفش رنگ

بخش پایینی و بالایی فیلم :

هم نور بخش قرمز شی و هم نور بخش آبی رنگ شی فقط به ناحیه وسط می‌رسند، در نتیجه در بخش بالایی و پایینی فیلم نمودی از رنگ‌های آبی و قرمز به چشم نمی‌خورند.



طبق شکل بالا تنها رنگ سبز است که به بخش بالایی و پایینی می‌رسد در نتیجه رنگ قسمت بالایی و پایینی فیلم سبز می‌باشد.

سوال 2

نوع دوربین لنز دار است ، { دلیل }

در دوربین pinhole ، کل تصویر به شکلی منظم نگاشته می شود روی پرده ای پشت عدسی ، در نتیجه اگر اندازه دریچه به درستی انتخاب شده باشد ، پرتو ها کمتر هستند و تنها از دریچه ای کوچک وارد می شوند ، در نتیجه سازمان یافته تر و متمرکر ترند و جایی در عکس تار نمی شود .

در نتیجه این دوربین لنزی است .

دلیل تار بودن بالا و پایین عکس بحث Depth of field است . هر چه شی از عدسی دورتر باشد ، تصویرش جلوتر از صفحه پشت عدسی واضح می شود و هر شی به عدسی نزدیک تر (از حدی نزدیکتر) باشد ، تصویر در جایی پشت صفحه پشت عدسی واضح می شود .

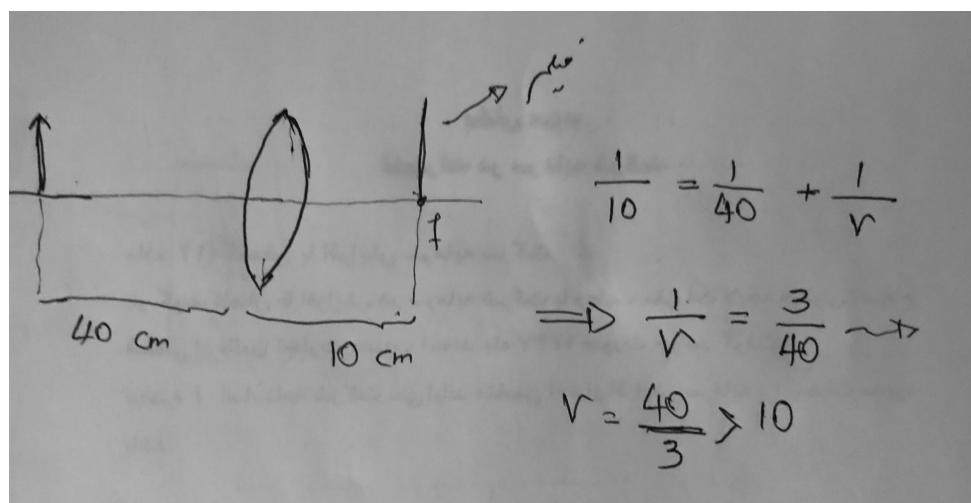
در هر دو صورت تصویر روی پرده ای پشت عدسی واضح نیست . در یک نقطه که تصویر تماماً روی فاصله کانونی میافتد تصویر واضح است و در غیر این صورت ، در صورتی تصویر واضح است که فاصله اش از آن نقطه از حدی بیشتر نباشد .

برای بهبود :

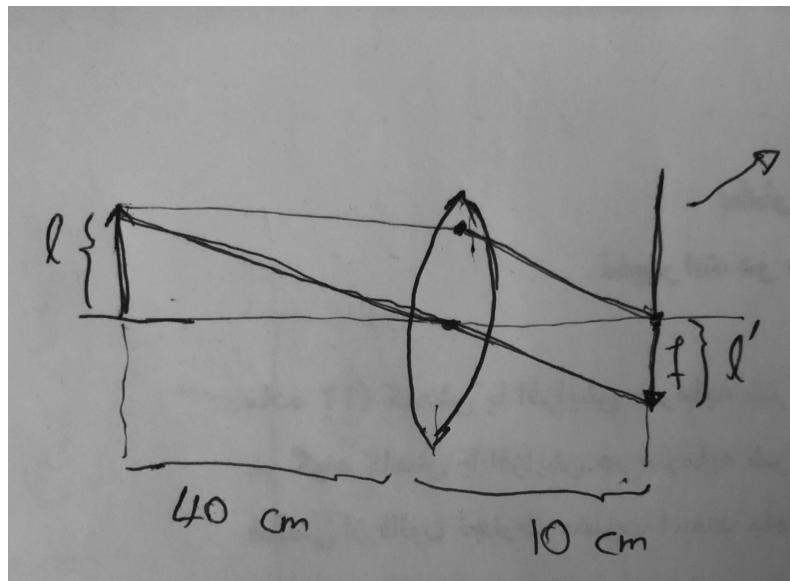
هم از دریچه استفاده می کنیم هم از عدسی . هر چه دریچه کوچکتر باشد ، از آنجایی که پرتو های اضافی حذف می شوند ، تاری کمتر مشاهده می شود .

سوال 3

همانطوری که در عکس و نوشته های زیر دیده می شود ، فاصله تصویر واضح از لنز بیشتر از فاصله فیلم از لنز است . در نتیجه تصویر به شکل واضح روی فیلم نمی افتد و تار است .



با رسم پرتو ها متوجه می شویم که دو مثلث متشابه وجود دارد که و در آن مثلث های متشابه رابطه زیر برقرار است ، در نتیجه طول تصویر هم کمتر خواهد بود (یک چهارم طول شی)



$$\frac{l}{l'} = \frac{40 \text{ cm}}{10 \text{ cm}} = 4 \rightarrow l = 4l'$$

برای گرفتن تصویر واضح تر می توانیم دو کار انجام دهیم :

1. بردن فیلم به جایی در فاصله 3/40 لنز
2. تغییر دادن قطر لنز به گونه ای که فاصله کانونی تغییر کند .

سوال 4

در متدهای آرگومان pattern داریم به نام `findChessBoardCorners` ، که به تعداد خانه های صفحه ی شطرنجی ربط دارد .

[منبع : opencv findChessboardCorners fails - Stack Overflow](#)

این متدهای خروجی دارد که یکی نشان دهنده موفقیت متدهای گذاری هاست و دیگری برای گوشه ها .

منبع : <https://gist.github.com/hackintoshrao/d32b9c43de7b2149facc2263920108d2>

متد cornerSubPix که برای اسکن گوشه ها با دقت بیشتر است ، چند آرگومان دریافت می کند . آرگومان اول یک عکس تک کاتاله است ، برای همین عکس را تبدیل به gray می کنیم .

همچنین متد drawChessBoardCorners نقاط گوشه را روی عکس اسکن شده نمایش می دهد . عکس خروجی را به کمک متد imwrite در عکس جدیدی ریختم :

کد نهایی :

```
import cv2
import os
import numpy as np

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# undistort with just one image
dirname = os.path.dirname(__file__)
path = os.path.join(dirname , 'images/img1.png' )

img = cv2.imread(path ,1)
print(img.shape)

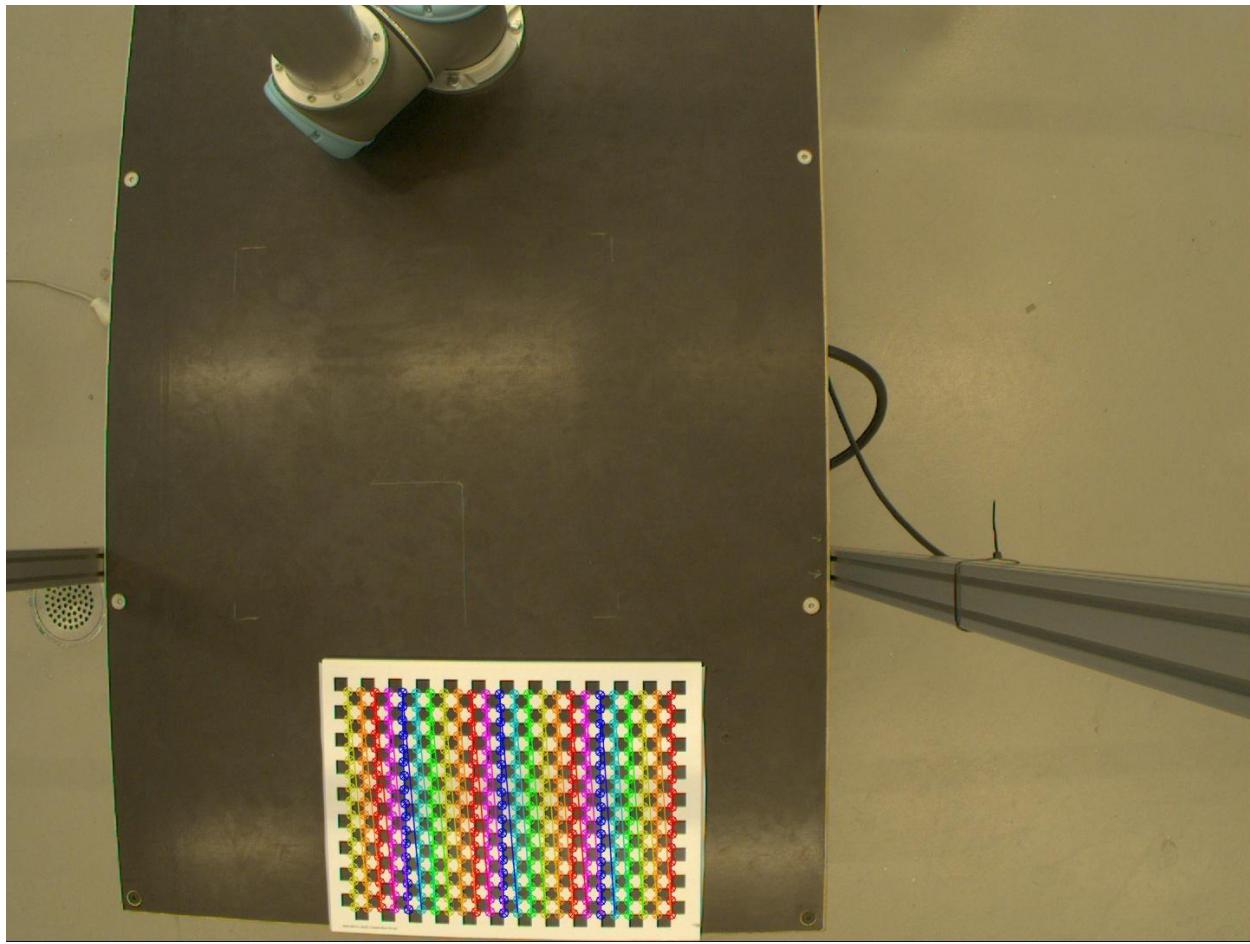
corner_count_x = 24
corner_count_y = 17

gray = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)

corners_found , corners = cv2.findChessboardCorners(gray , (corner_count_y,corner_count_x) , None )

if corners_found :
    improved_corners = cv2.cornerSubPix(gray , corners , (11,11), (-1,-1) , criteria )
    cv2.drawChessboardCorners(img , (corner_count_y , corner_count_x) , improved_corners , corners_found)
    cv2.imwrite("image_with_corners.jpg", img)
else :
    print("Corners were not successfully found .")
```

نتیجه به شکل زیر شد (عکس پیوست شده است) :



منبع : <https://learnopencv.com/camera-calibration-using-opencv/>

برای استخراج پارامتر ها از لینک زیر کمک گرفتم :

https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

طبق لینک بالا ، خروجی سوم متد calibrateCamera ضرایب اعوجاج که 5 تا هستند را به ترتیب نمایش می دهد . از آنها در کنسول پرینت گرفتیم و به شکل زیر شد :

```
dist_coeffs respectively k1 , k2 , p1 , p2 , k3:  
[[ -0.30733764  0.23906645 -0.0018685   0.0087472  -0.1604419 ]]
```

پارامتر های اعوجاج مقادیر زیر را دارد :

$$K_1 = -0.3$$

$$K_2 = 0.23$$

$$P_1 = -0.0018$$

$$P_2 = 0.0087472$$

$$K_3 = -0.16$$

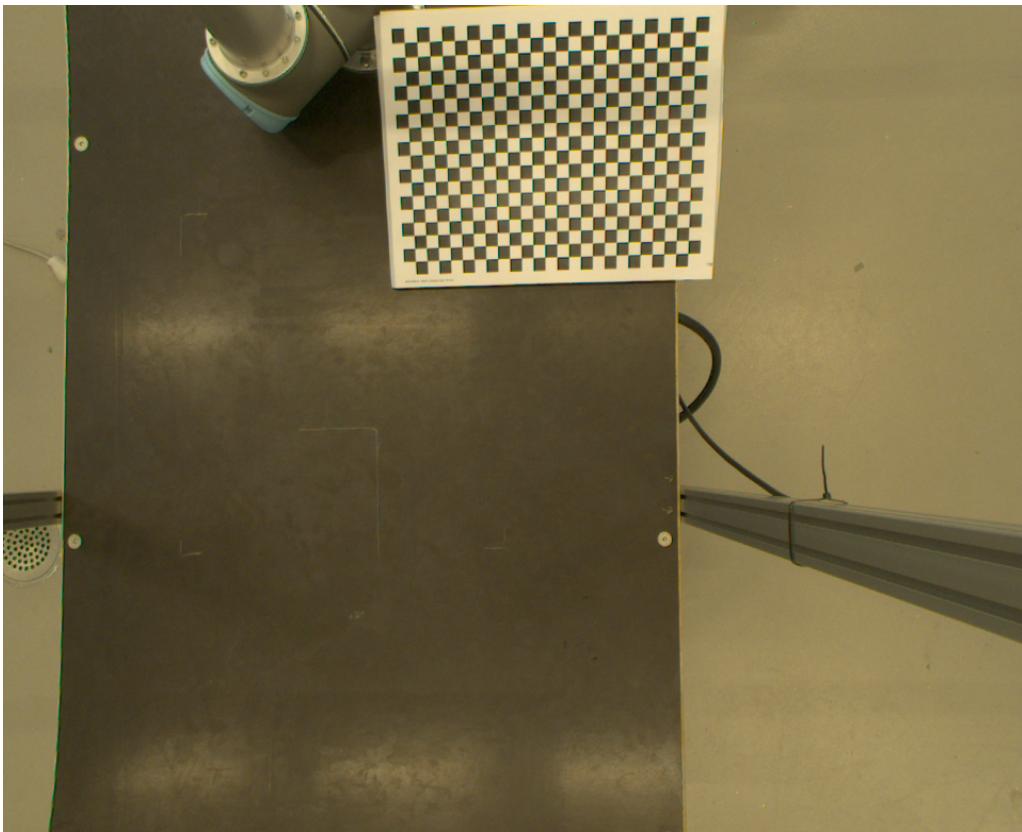
برای رفع اعوجاج از متده استفاده می کنیم . این متده به چند ورودی احتیاج دارد . عکس ، ماتریس دوربین و ماتریس بهینه دوربین و ضرایب اعوجاج دسته ای از آنها هستند .

برای به دست آوردن ضرایب خالص شده ای اعوجاج از متده استفاده می کنیم . این متده علاوه بر ضرایب ، به ما مختصات لازم برای کراپ کردن عکس (عکس رفع اعوجاج شده) هم می دهد . (roi)

منبع استفاده از متده است : cv2.getOptimalNewCameraMatrix

https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

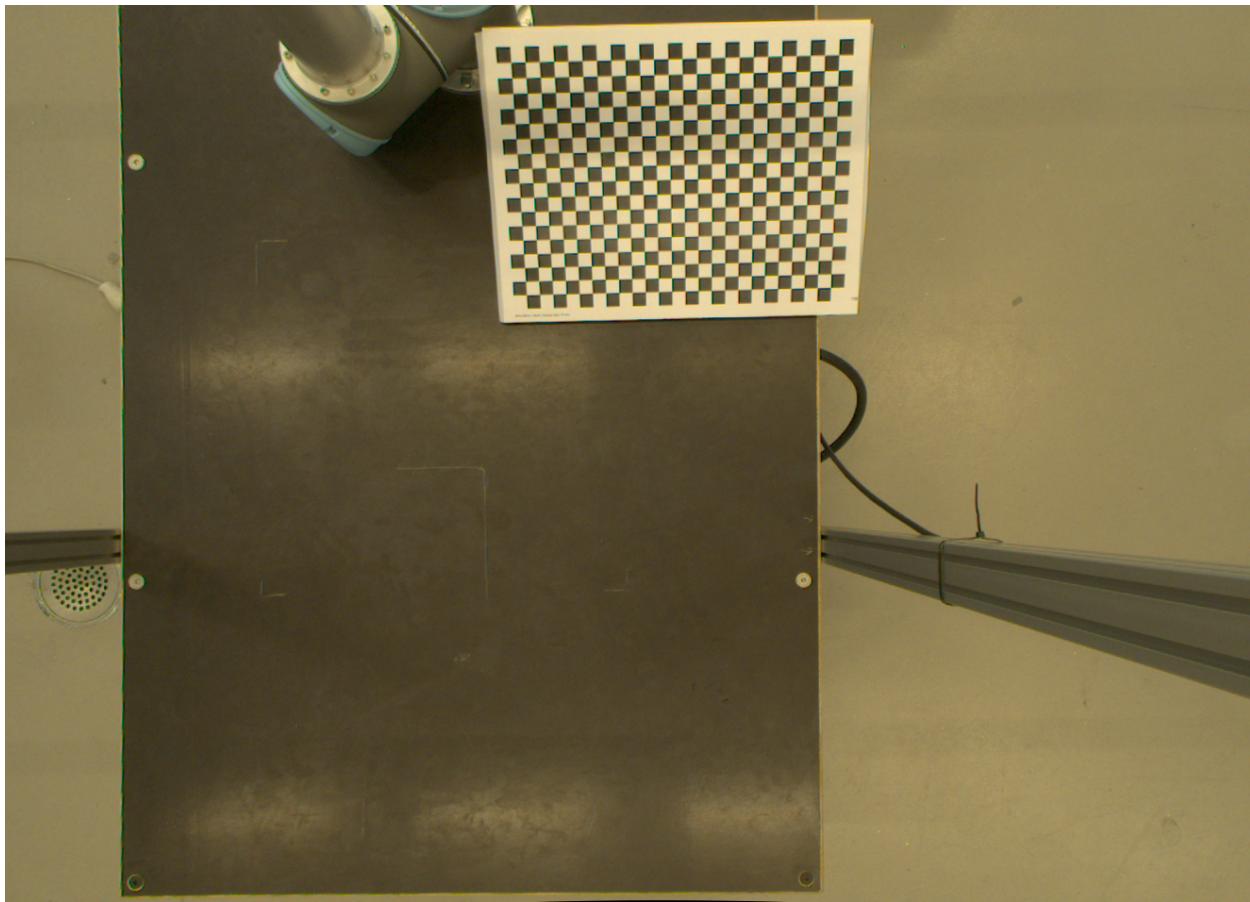
نتیجه رفع اعوجاج تنها با استفاده از عکس img1 به شکل زیر شد :



حال همین مراحل را تکرار میکنیم تنها با این تفاوت که از 4 عکس اول استفاده می کنیم .

طبعاً کد تغییر زیادی نمی کند . کد جدید تنها چند فایل بیشتر را می خواند و کارهای قبلی را روی آن انجام می دهد . برای همین تنها یک متده می نویسیم و برای آن چند خطی که متفاوت است ، یک متده به این متده بزرگ پاس می دهیم . کد ضمیمه شده است .

نتیجه (عکس رفع اعوجاج شده) در اثر خواندن از 4 فایل به شکل زیر است :



در اثر مقایسه چشمی این عکس ها متوجه می شویم که در ضلع پایین و سمت راست عکس دوم ، انحنای بیشتری می بینیم به قدری که مقداری سیاهی کمی در عکس دیده می شود . همچنین تفاوتی در مقادیر پارامتر های اعوجاج هم دیده می شود .

```
dist_coeffs respectively k1 , k2 , p1 , p2 , k3:  
[[-0.30733764  0.23906645 -0.0018685   0.0087472  -0.1604419 ]]
```

```
dist_coeffs respectively k1 , k2 , p1 , p2 , k3:  
[[-0.2057339   0.19798757 -0.00082691 -0.01646572 -0.17887078]]
```