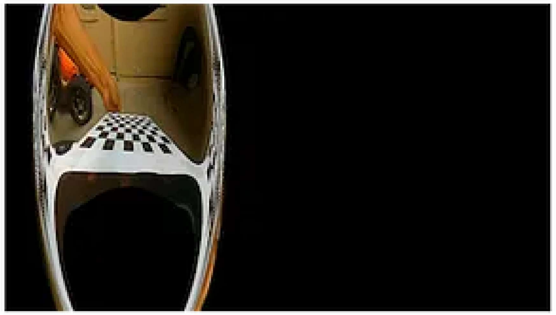
Calibrate fisheye lens using OpenCV — part 1

Kenneth Jiang Follow 4 min read · Sep 29, 2017

When you are using a fisheye (>160 degree field-of-view) lens, the 'classic' way in OpenCV to calibrate lens may not work for you. Even if you carefully follow steps in OpenCV document, you may end up with 'undistorted' images like these:



If you find yourself in similar situtation, read on.

Since version 3.0 OpenCV has included a package `cv2.fisheye` that decently handles fisheye lens calibration. However that module does not have easy-to-follow tutorials for readers who are not a mathematician.

That's why I wrote this tutorial. My goal is to get your fisheye lens calibrated even if you don't have any prior experience in OpenCV (or optical science for that matter).

At 30000 feet, calibrating your lens involves 2 steps.

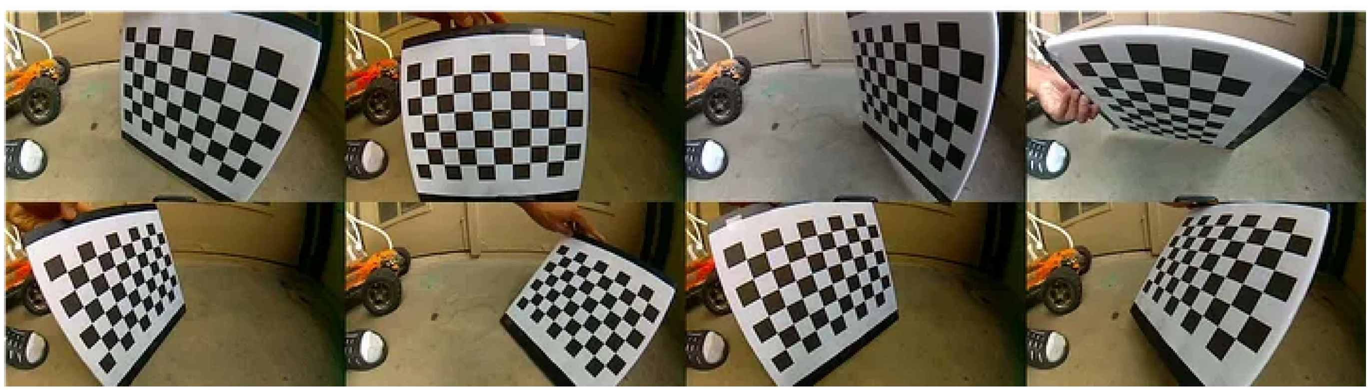
1. Help OpenCV find 2 parameters intrinsic to your lens. OpenCV call them K and D. You don't really need to know what K and D are except they are numpy arrays.

2. Un-distort images by applying K and D.

Find K and D

- Download the checkerboard pattern and print it on a paper (letter or A4 size). You also want to attach the paper to a hard, flat surface such as a piece of cardboard. The key here: **straight lines need to be straight.**

- Hold the pattern in front of your camera and capture some images. You want to hold the pattern in different positions and angles. The key here: the patterns need to appear distorted **in a different ways** (so that OpenCV knows as much about your lens as possible). Some examples are:



- Save these pictures in a folder in JPG format.
- Now the fun part — write some code. Not really. You just need to copy this piece of Python script to a file creatively named `calibrate.py` in the folder where you saved these images earlier.

```
import cv2
assert cv2.__version__[0] == '3', 'The fisheye module requires opencv
version >= 3.0.0'

import numpy as np
import os
import glob

CHECKERBOARD = (6,9)

subpix_criteria = (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER,
30, 0.1)
calibration_flags =
cv2.fisheye.CALIB_RECOMPUTE_EXTRINSIC+cv2.fisheye.CALIB_CHECK_COND+cv
2.fisheye.CALIB_FIX_SKEW

objp = np.zeros((1, CHECKERBOARD[0]*CHECKERBOARD[1], 3), np.float32)
objp[0,:,:2] = np.mgrid[0:CHECKERBOARD[0],
0:CHECKERBOARD[1]].T.reshape(-1, 2)

_img_shape = None
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

images = glob.glob('*.jpg')

for fname in images:
    img = cv2.imread(fname)
    if _img_shape == None:
        _img_shape = img.shape[:2]
```

```
    else:
        assert _img_shape == img.shape[:2], "All images must share
the same size."

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
cv2.CALIB_CB_ADAPTIVE_THRESH+cv2.CALIB_CB_FAST_CHECK+cv2.CALIB_CB_NOR
MALIZE_IMAGE)
    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)
        cv2.cornerSubPix(gray,corners,(3,3),(-1,-1),subpix_criteria)
        imgpoints.append(corners)

N_OK = len(objpoints)
K = np.zeros((3, 3))
D = np.zeros((4, 1))
rvecs = [np.zeros((1, 1, 3), dtype=np.float64) for i in range(N_OK)]
tvecs = [np.zeros((1, 1, 3), dtype=np.float64) for i in range(N_OK)]
rms, _, _, _, _ = \
    cv2.fisheye.calibrate(
        objpoints,
        imgpoints,
        gray.shape[::-1],
        K,
        D,
        rvecs,
        tvecs,
        calibration_flags,
        (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER, 30, 1e-6)
    )
print("Found " + str(N_OK) + " valid images for calibration")
print("DIM=" + str(_img_shape[::-1]))
print("K=np.array(" + str(K.tolist()) + ")")
print("D=np.array(" + str(D.tolist()) + ")")
```

- Run `python calibrate.py` . If everything goes smoothly the script will print out something like this:

```
Found 36 images for calibration
DIM=(1600, 1200)
K=np.array([[781.3524863867165, 0.0, 794.7118000552183], [0.0,
779.5071163774452, 561.3314451453386], [0.0, 0.0, 1.0]])
D=np.array([[-0.042595202508066574], [0.031307765215775184],
[-0.04104704724832258], [0.015343014605793324]])
```

Un-distort images

Once you obtain K and D, follow the steps here to un-distort images if:

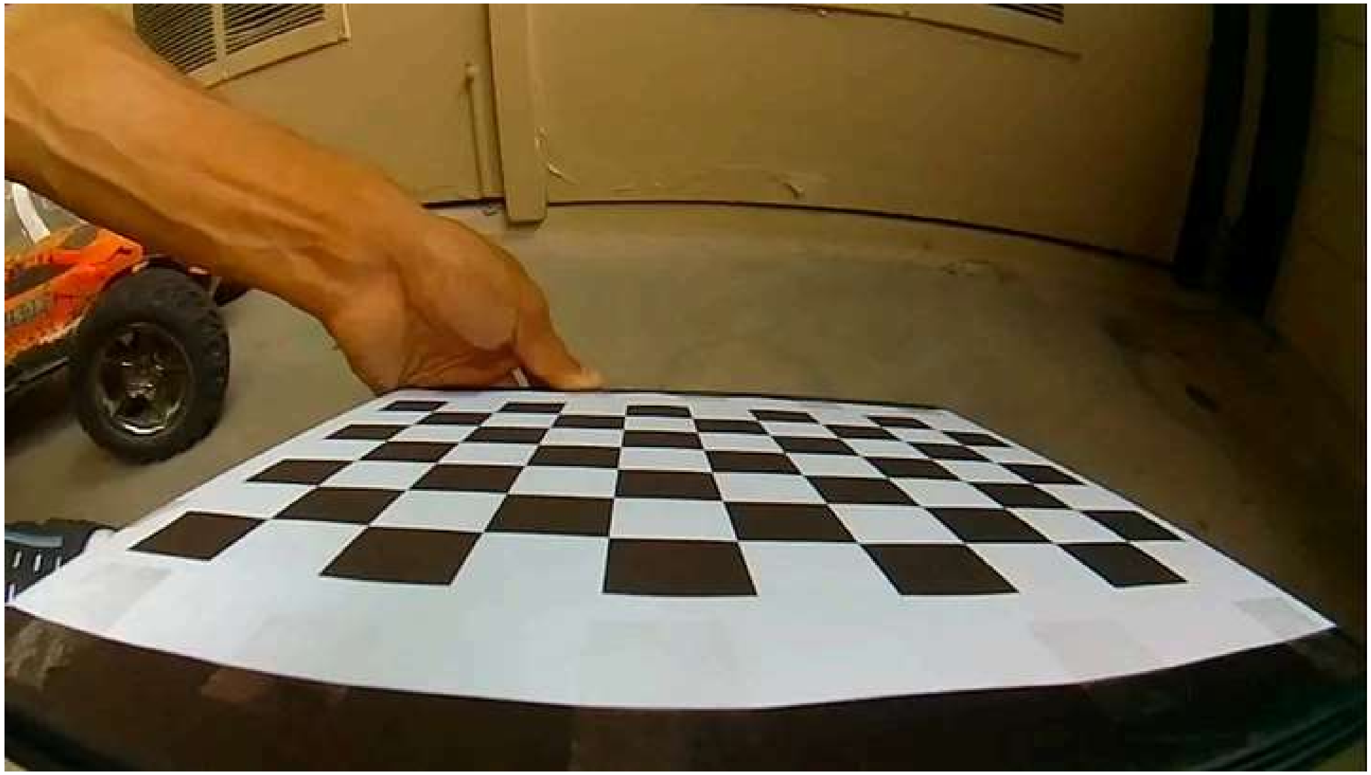
- The images you need to undistort are the same dimension as the ones captured during calibration.
- You are ok with some areas around the edge being cropped out to keep the un-distorted images neat.

Your life will be a lot easier if this is the case. Otherwise you need to follow part 2 of the tutorial.

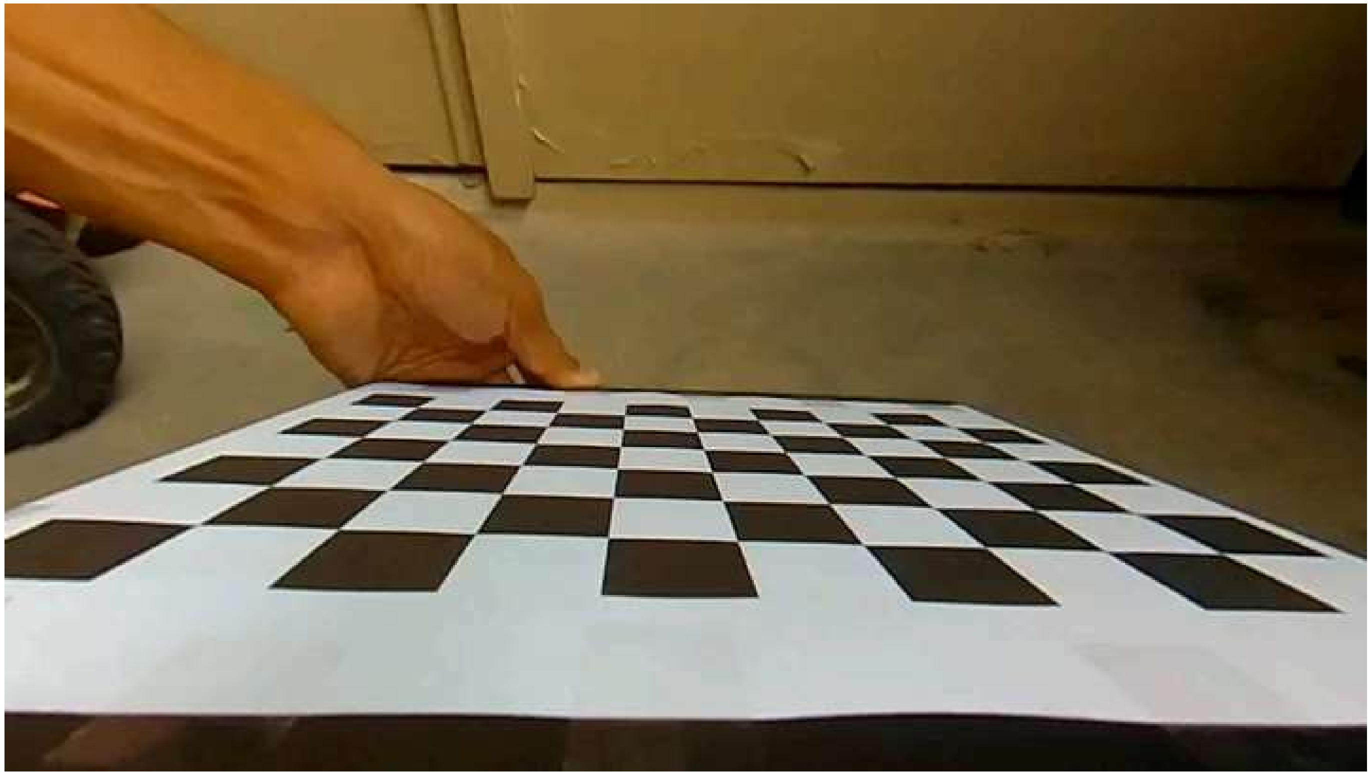
Create file `undistort.py` with the following python code:

```
# You should replace these 3 lines with the output in calibration
step
DIM=XXX
K=np.array(YYY)
D=np.array(ZZZ)

def undistort(img_path):

    img = cv2.imread(img_path)
    h,w = img.shape[:2]

    map1, map2 = cv2.fisheye.initUndistortRectifyMap(K, D, np.eye(3),
K, DIM, cv2.CV_16SC2)
    undistorted_img = cv2.remap(img, map1, map2,
interpolation=cv2.INTER_LINEAR, borderMode=cv2.BORDER_CONSTANT)

    cv2.imshow("undistorted", undistorted_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == '__main__':
    for p in sys.argv[1:]:
        undistort(p)
```

Now run `python undistort.py file_to_undistort.jpg` and that's it.



Before



After

If you look closely, you may notice a problem: a significant chunk in the original image gets cropped out in the process. For instance, the orange RC car to the left side of the image only has half a wheel kept in the undistorted image. As a matter of fact about 30% of the pixels in original image get lost. Ouch!

If you want to have a way to get back those lost pixels, continue to part 2 of this tutorial.

Python Opencv Fisheye Cameras Lenses

Written by Kenneth Jiang

125 followers · 16 following

Follow

Responses (35)

Write a response

What are your thoughts?

Michael Corleone
Feb 28, 2018

Hi Kenneth, great article!

I have a question: while running your code and making sure that the points are correctly found on the calibration images, I get this error:

28 1 reply Reply

Sid Rajan
Aug 7, 2018

OpenCV(3.4.2) C:\projects\opencv-python\opencv\modules\calib3d\src\fisheye.cpp:711: error: (-215:Assertion failed) !objectPoints.empty() && !imagePoints.empty() && objectPoints.total() == imagePoints.total() in function 'cv::fisheye::calibrate

Why am I getting this error. Can't find much online either>

3 1 reply Reply

Abhishek Gupta
Oct 31, 2018

Can someone please explain or give brief information on the four values in Array D, which we got in the output. I just want to know what each value means in the calibration world.

Thank You!

5 Reply

See all responses