Calibrate fisheye lens using OpenCV—part 2

medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-part-2-13990f1b157f

September 29, 2017

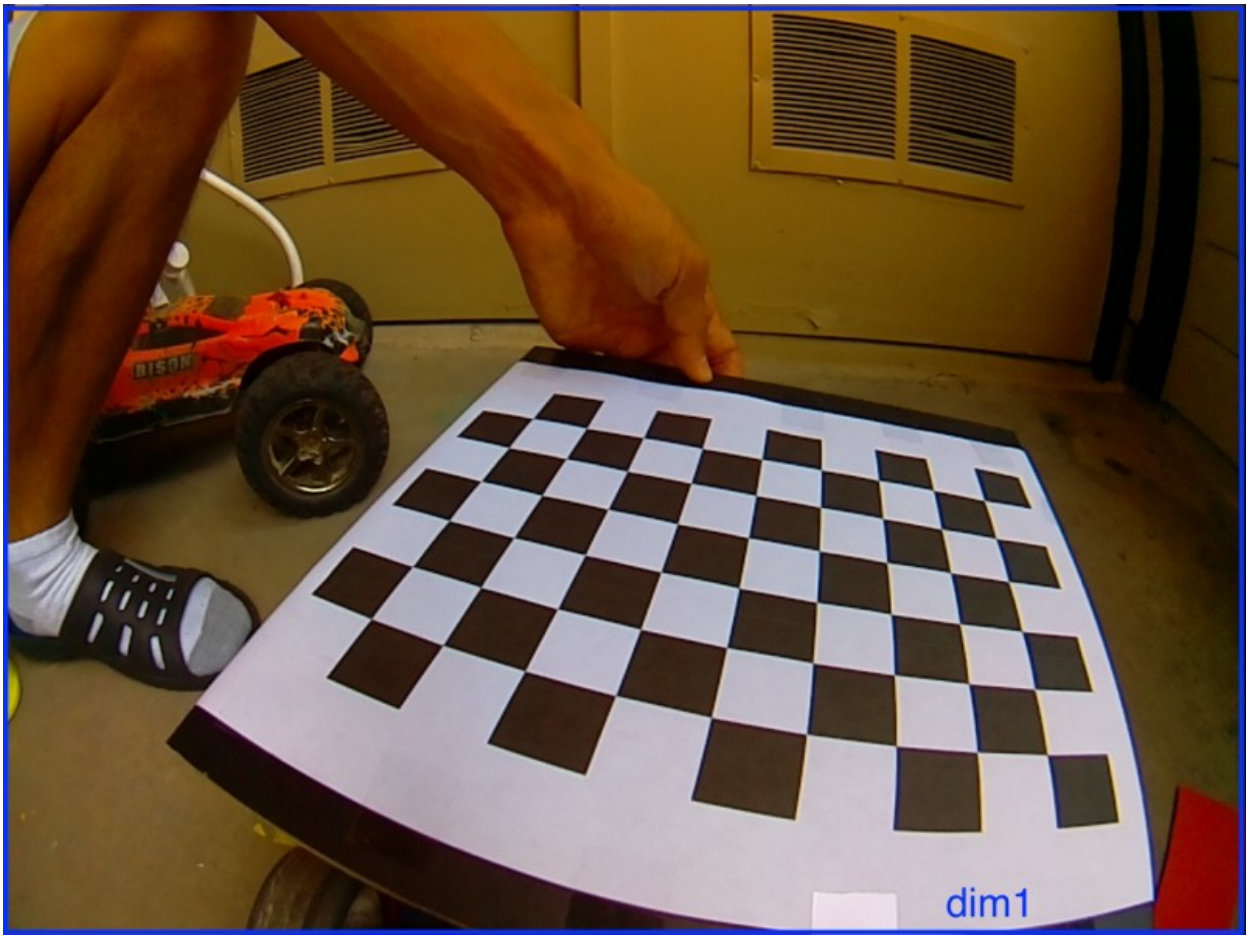
In part 1 we covered some basics on how to use OpenCV to calibrate your fisheye lens. If you are happy with the un-distorted image after following part 1—great, you don't have to read or understand any of the things I talk about in part 2.

However, you will have to bite the bullet to understand some nitty-gritty details if you run any of these situations:

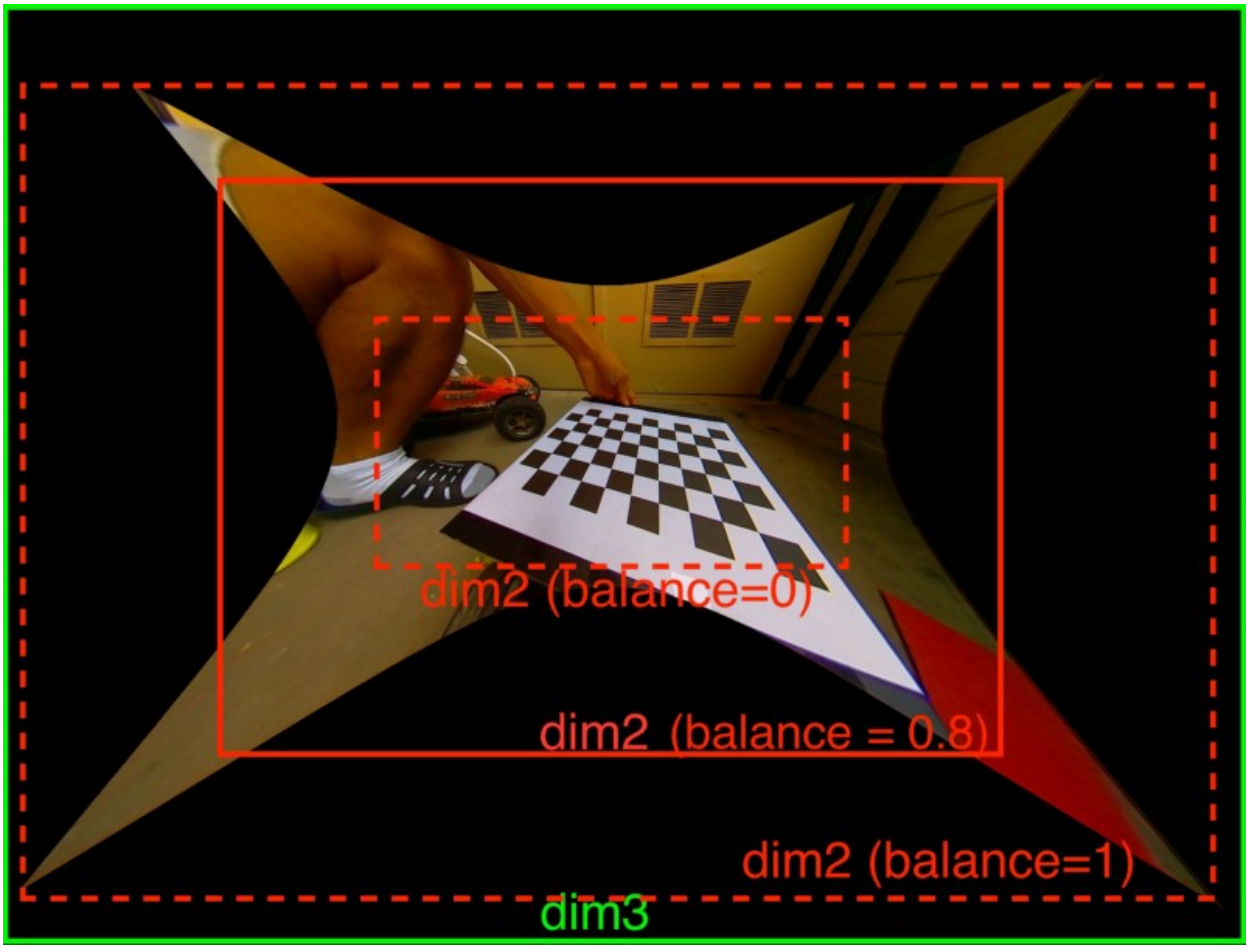
- the images to un-distort have different dimensions than the ones used in calibration;
- the amount of pixels that get cropped off by the default un-distortion settings is too big to be acceptable.

We will need to start with understanding what 'balance' (also called 'alpha' in classic `cv2` module) is, and how it impacts the dimension and shape of the undistorted image.

First of all, you need to understand the shape of an image will change after being un-distorted. It may be a good idea to take a diversion to this SO post if you are not quite sure.



Before



Note: As you can tell, OpenCV `fisheye` module over-compensated the corners by stretching them too far. But it did a decent job for the majority of the image.

There are 3 different dimensions involved here:

- `dim1`. Dimension of the original image.
- `dim2`. This is the dimension of the box you want to keep after un-distorting the image. `dim2` is tricky to understand because it's influced by `balance`, which basically tells OpenCV how much of the image you want to keep. When `balance = 0`,OpenCV will keep the best part of the image for you. Whereas `balance = 1` tells OpenCV to keep every single pixel of the original image, which means a lot of black filling areas and overstretched corners. Please note that `dim2` needs to have the same aspect ratio as `dim1`.
- `dim3`. Dimension of the final box where OpenCV will put the undistorted image. It can be any size and any aspect ratio. But most of the times you probably want to make it the same as `dim2` unless you want to do some cropping to the un-distorted image.

Whew! I hope I made it easier for you to understand than the official OpenCV docs. Assuming I did a good job explaining this, you should be able to know what `dim1`, `dim2`, `dim3` and `balance` are for your application.

Pass `dim2`, `dim3` and `balance` (`dim1` can be derived from input image) to `undistort` method below and you should have your image un-distorted the way you want it!

```
# 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, balance=0.0, dim2=None, dim3=None):

    img = cv2.imread(img_path)
    dim1 = img.shape[:2][::-1]  #dim1 is the dimension of input image to un-distort

    assert dim1[0]/dim1[1] == DIM[0]/DIM[1], "Image to undistort needs to have same
aspect ratio as the ones used in calibration"

    if not dim2:
        dim2 = dim1

    if not dim3:
        dim3 = dim1

    scaled_K = K * dim1[0] / DIM[0]  # The values of K is to scale with image
dimension.
    scaled_K[2][2] = 1.0  # Except that K[2][2] is always 1.0

    # This is how scaled_K, dim2 and balance are used to determine the final K used
to un-distort image. OpenCV document failed to make this clear!
    new_K = cv2.fisheye.estimateNewCameraMatrixForUndistortRectify(scaled_K, D,
dim2, np.eye(3), balance=balance)
    map1, map2 = cv2.fisheye.initUndistortRectifyMap(scaled_K, D, np.eye(3), new_K,
dim3, 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)
```

Of course you should experiment with different values to see how they change the final image. At the minimum you want to play with `balance` because it's so much fun!