# HW3:

## 1 作业要求

# HW#3: Camera Calibration

### 摄像机标定及俯瞰视角变换：

**Input:**

**1.**一组关于棋盘的图像文件用于标定；

**2.**同一相机拍摄图像的俯瞰视角变换 **view.jpg**；

**3.**测试图像需要自己拍摄，并同作业一起提交。**(**可用**OpenCV**样本数据对自己程序进行测试与验证**)**

**Output:**

**1.**将摄像机标定后的参数输出；

**2.**显示主要的中间步骤，包括棋盘角点检测结果、镜头畸变校正的结果。如果选用了两幅以上的图像，只输出其中两幅。（如用**OpenCV**的**cvShowImage**显示）；

**3.**显示并保存输出俯瞰视角变换后的图像 **birdseye-view.jpg**。

**Hint: (读《Learning OpenCV》 第11、12两章， pdf及源码可在钉钉课程群下载)**

**1.**精读**Chapter11**的两段源代码： **ch11_ex11_1.cpp ch11_ex11_1_fromdisk.cpp**；

**2.**根据**1**中的代码，划分功能模块，完成自己的相机标定代码；

**3.**精读**Chapter12**的一段源代码：**ch12_ex12_1.cpp**；

**4.**将上述两份源代码功能合在一起。

(下页有示意)

**提交截至时间：2025年1月5日 8:00**



## 2 运行

### 2.1 环境

```
python==3.9.18
opencv-python==4.10.0.84
numpy==1.26.4
```

### 2.2 运行

```
cd hw3
python main.py
```

# 3 实现

## 3.1 数据来源

- iphone 15前置摄像头，拍摄电脑展示图片

## 3.2 相机标定

- 主要流程如下

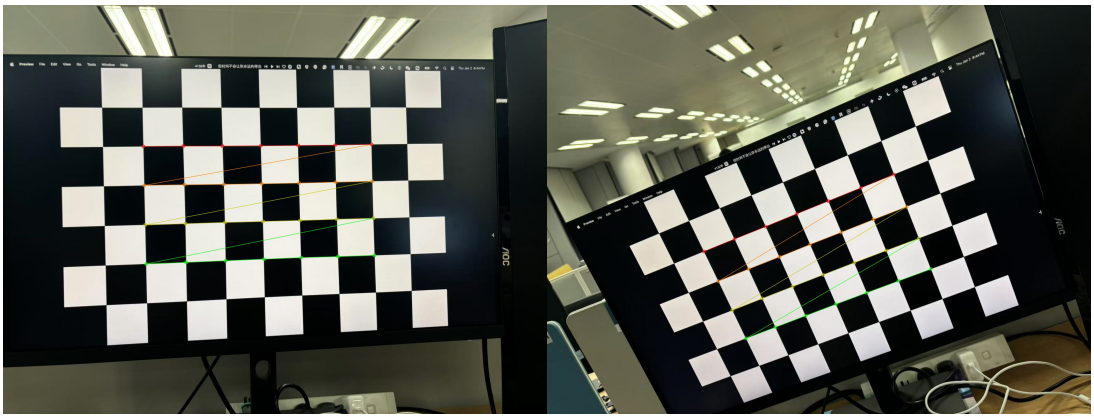
```
img = cv2.imread(image_path)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
size = gray.shape[::-1]
ret, corners = self.get_corner(gray)
if ret:
    obj_points.append(self.objp)
    img_points.append(corners)
    cv2.drawChessboardCorners(img, self.board_size, corners, ret)
    if save:
        save_path_i = os.path.join(self.output_path, 'corners_'+str(i)+'.jpg')
        cv2.imwrite(save_path_i, img)
        print("Saving image: ", save_path_i)
```

- 得到相机参数

```
Camera matrix:  [[1.27338517e+03 0.00000000e+00 8.60161638e+02]
 [0.00000000e+00 1.27334847e+03 6.39698502e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion coefficients:  [[ 2.87738580e-01 -1.64289178e+00  1.74713587e-04
 1.46416301e-04
   2.60959893e+00]]
Rotation vectors:  (array([[ 0.05661588],
       [-0.06887514],
       [-0.03784882]]), array([[-0.0115898 ],
       [-0.10066124],
       [-0.02593096]]), array([[ 0.18677958],
       [ 0.07713145],
       [-0.39083471]]), array([[ 0.04883779],
       [ 0.16379245],
       [-0.07388073]]), array([[-0.2543957 ],
       [-0.08430534],
       [-0.03456773]]), array([[-0.06215935],
       [ 0.23680063],
       [-0.38988796]]), array([[ 0.17659006],
       [-0.26566874],
       [-0.01166437]]), array([[ 0.23699089],
       [-0.13052591],
       [-0.01316686]]), array([[ 0.00220345],
       [-0.29225638],
```

```
        [ 0.0234284 ]]), array([[-0.14575027],
        [-0.30785823],
        [ 0.17841207]]))
Translation vectors:  (array([[-2.88160559],
        [-3.82978407],
        [15.50593771]]), array([[-3.4330396 ],
        [-1.61083398],
        [10.54266537]]), array([[-3.48007969],
        [ 1.23969997],
        [12.20716971]]), array([[-3.46957103],
        [-1.16731658],
        [11.69662273]]), array([[-3.06211859],
        [-1.6459245 ],
        [ 9.95587667]]), array([[-3.94643561],
        [ 0.14085341],
        [11.55656299]]), array([[-1.30365844],
        [-0.57443397],
        [10.56071718]]), array([[-2.66640978],
        [-0.9829703 ],
        [10.64275398]]), array([[-1.52863813],
        [-1.42046782],
        [ 9.68438769]]), array([[-1.0763635 ],
        [-2.07618703],
        [ 9.62378177]]))
```

- 展示两张角点检测图

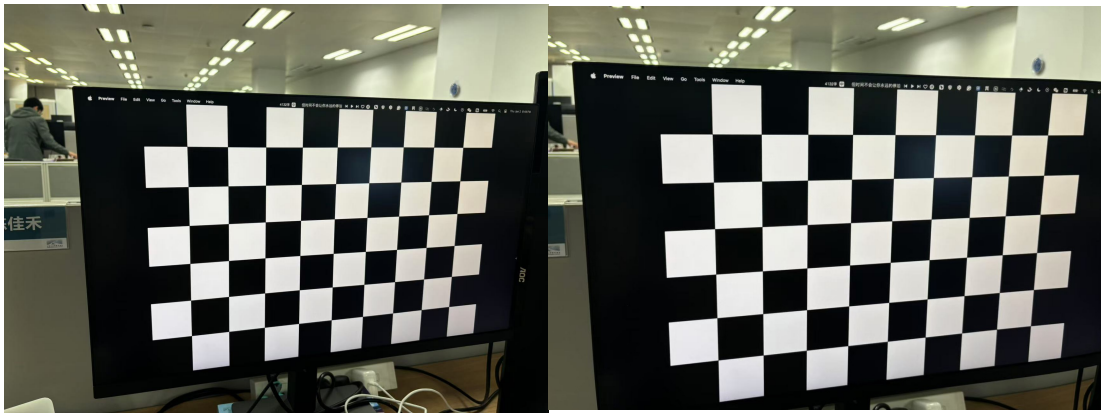


## 3.3 镜头畸变校正

- 核心代码如下

```
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(self.mtx, self.dist, (w, h), 1,
(w, h))
dst = cv2.undistort(img, self.mtx, self.dist, None, newcameramtx)
```

- 右图为校正结果



## 3.4 bev图获取

- 核心代码如下，获取鸟瞰图的transform后apply到图像

```
objp = np.float32([[0,0], [h-1,0], [0,w-1], [h-1,w-1]]) * 150
imgp = np.float32([corners[0], corners[h-1], corners[-h], corners[-1]])
H = cv2.getPerspectiveTransform(imgp, objp)
bev = cv2.warpPerspective(img, H, img.shape[:2][::-1])
```

- 右图为bev图