**实验报告**

专业：计算机科学与技术

姓名：贺情怡

学号：3180105438

日期：2021/1/7

课程名称： 计算机视觉 指导老师： 宋明黎 成绩：

实验名称： HW#4:相机标定正畸并生成鸟瞰图

**一、实验目的和要求**

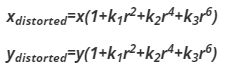
1. 参考Learning OpenCV示例18-1，利用棋盘格图像进行相机定标，将参数写入xml文件保存。棋盘格图像见群文件Learning OpenCV/LearningOpenCV\_Code/LearningOpenCV\_Code/calibration

2. 参考示例19-1，根据求得的内参实现鸟瞰图（俯视）转换，测试图片见群文件Learning OpenCV/LearningOpenCV\_Code/LearningOpenCV\_Code/birdseye

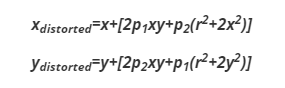
**二、实验内容和原理**

**相机参数标定**

参考OpenCV官方给出的文档（<https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html>），可以看到它给出的理论知识。对于畸变，OpenCV考虑了径向和切向因素。径向畸变的存在表现为“桶”或“鱼眼”效应，对于径向畸变使用以下公式正畸：



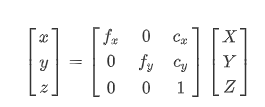
由于成像透镜与成像平面不会完全平行，所以会产生切向畸变。可通过以下公式表示：



因此，我们有五个失真参数，在OpenCV中表示为一行五列矩阵：



摄像机矩阵中包括了光学中心坐标和焦距，可以用它来将世界坐标转换为像素坐标：



坐标是以齐次的方式写的，其中Z=z=1。

**鸟瞰视角**

本次实验中要求的鸟瞰视角是利用标定时得到的棋盘格坐标与理想中棋盘格应有的坐标一一对应来完成视角转换得到的。

利用得到的图像中的棋盘格的四个角的坐标和理想中的(0, 0), (0, 8), (5, 0), (5, 8)，可以得到一个映射矩阵，再利用这个矩阵处理照片，即可得到照片中每个像素点映射到以棋盘格为标准坐标系的空间中对应的位置。

**三、实验步骤与分析**

**读入图片并标注棋盘格**。

利用了glob来读入当前路径下的图片，使用cv2自带的函数进行棋盘格的识别（6\*9个内部角点）。（因此在esc时不会直接退出而会继续处理下一张图像）

由于一张图用来求相机参数就基本够了，所以没有对所有照片求参再合并。

|  |
| --- |
| img = cv2.imread(fname)  img = cv2.resize(img, (0, 0), fx = 0.5, fy = 0.5)  gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  cv2.imshow('img', img)  cv2.waitKey(1)  plt.subplot(2,2,1), plt.title('original image')  plt.imshow(img)  print('Done\n')  # Find the chess board corners  print('\nFinding Chessboard...')  print('-----------------------------------------')  ret, corners = cv2.findChessboardCorners(gray, (9,6),None)  # If found, add object points, image points (after refining them)  if ret == True:  objpoints.append(objp)  corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)  imgpoints.append(corners2)  img\_ = img.copy()  # Draw and display the corners  # Use img\_ here instead of img is for the later use of undistorted image  img\_ = cv2.drawChessboardCorners(img\_, (9,6), corners2,ret)  cv2.imshow('img',img)  cv2.waitKey(1) |

**正畸**

利用cv2给出的calibrateCamera函数，可以获得相机的参数矩阵和图像畸变的具体参数

|  |
| --- |
| # Using the cv2.calibrateCamera function, it can generate the needed matrices for the camera pose and how the image is distorted  ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None) |

将参数保存至matrices.xml中及coefs.npz中（后者方便python程序直接使用）

|  |
| --- |
| # serialize the matrices  np.savez('coefs.npz', mtx=mtx, dist=dist, rvecs=rvecs, tvecs=tvecs)  cv\_file = cv2.FileStorage("./save/matrices.xml", cv2.FILE\_STORAGE\_WRITE)  cv\_file.write("mtx", mtx)  cv\_file.write("dist", dist)  cv\_file.write("rvecs", rvecs[0])  cv\_file.write("tvecs", tvecs[0])  cv\_file.release() |

找到x和y方向上像素点的畸变函数，再将像素点映射到校正后的位置上

|  |
| --- |
| # undistort  h, w = img.shape[:2]  # generate the new matrix of camera pose after undistortion  newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))  # here we can get how the pixels are mapped to the new image  mapx,mapy = cv2.initUndistortRectifyMap(mtx,dist,None,newcameramtx,(w,h),5)  dst = cv2.remap(img,mapx,mapy,cv2.INTER\_LINEAR) |

得到剪切后的图像

|  |
| --- |
| x,y,w,h = roi  cropped = dst[y:y+h, x:x+w] |

**生成鸟瞰图**

使用校正后的图像来重新标定棋盘格的位置，获取鸟瞰图的视角变换矩阵。

|  |
| --- |
| # use the undistorted image as the source, find the corners for generate the birdview image  img = dst  gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  ret, corners = cv2.findChessboardCorners(gray, (9,6),None)  print('ret = ', ret)  if ret == True:  objpoints.append(objp)  corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)  imgpoints.append(corners2)  img\_ = img.copy()  # Draw and display the corners  img\_ = cv2.circle(img\_, (corners[0][0][0], corners[0][0][1]), 63, (0, 0, 255), 5)  img\_ = cv2.circle(img\_, (corners[8][0][0], corners[8][0][1]), 63, (0, 255, 0), 5)  img\_ = cv2.circle(img\_, (corners[45][0][0], corners[45][0][1]), 63, (255, 255, 0), 5)  img\_ = cv2.circle(img\_, (corners[53][0][0], corners[53][0][1]), 63, (255, 0, 255), 5)  img\_ = cv2.drawChessboardCorners(img\_, (9,6), corners2,ret)  cv2.imshow('img',img\_)  cv2.waitKey(1)  corners = np.squeeze(corners) |

进行50x的缩放，防止生成的图像太小。

|  |
| --- |
| Scale = 50  objpts = np.float32([[0, 0], [8, 0], [0, 5], [8, 5]])  objpts = objpts \* Scale  print(objpts)  imgpts = np.float32([corners[0],corners[8],corners[45],corners[53]])  print(imgpts)  cv2.destroyAllWindows()  # find how it transformed  H = cv2.getPerspectiveTransform(imgpts, objpts) |

显示鸟瞰图，并进行键盘动作的处理。

|  |
| --- |
| print("\nPress 'd' for lower birdseye view, and 'u' for higher (it adjusts the apparent 'Z' height), Esc to exit")  Z = H[2, 2]  while True:  H[2, 2] = Z  Perspective\_img = cv2.warpPerspective(img, H, (img.shape[1], img.shape[0]))  cv2.imshow("Birdseye View", Perspective\_img)  KEY = cv2.waitKey() & 0xFF  if KEY == ord('u'):  Z += 0.05  if KEY == ord('d'):  Z -= 0.05  if KEY == 27:  cv2.destroyAllWindows()  exit() |

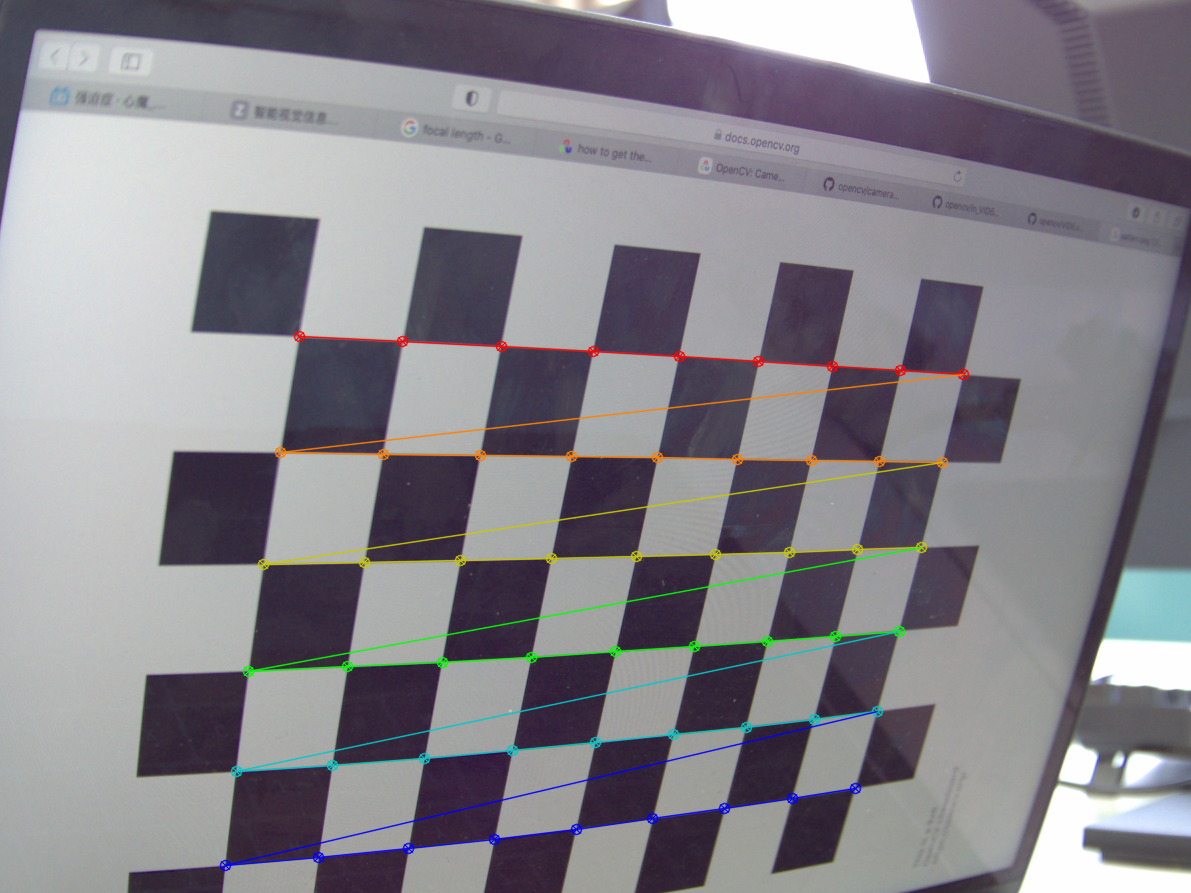
**四、实验结果**

1. **自己拍摄的照片**

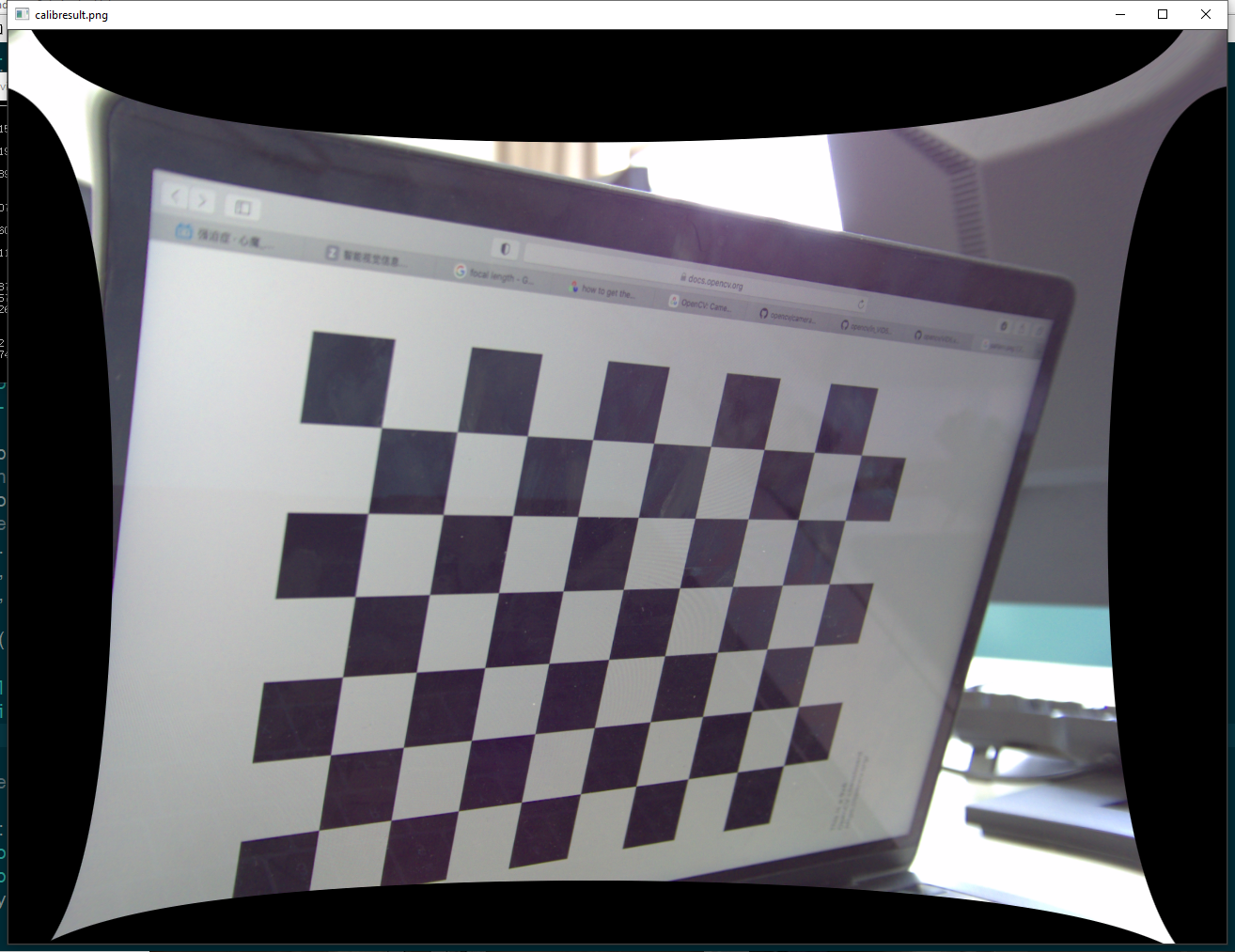
由于给出的图畸变不是很明显，直接使用了自己拍摄的照片来测试正畸效果。

**正畸**

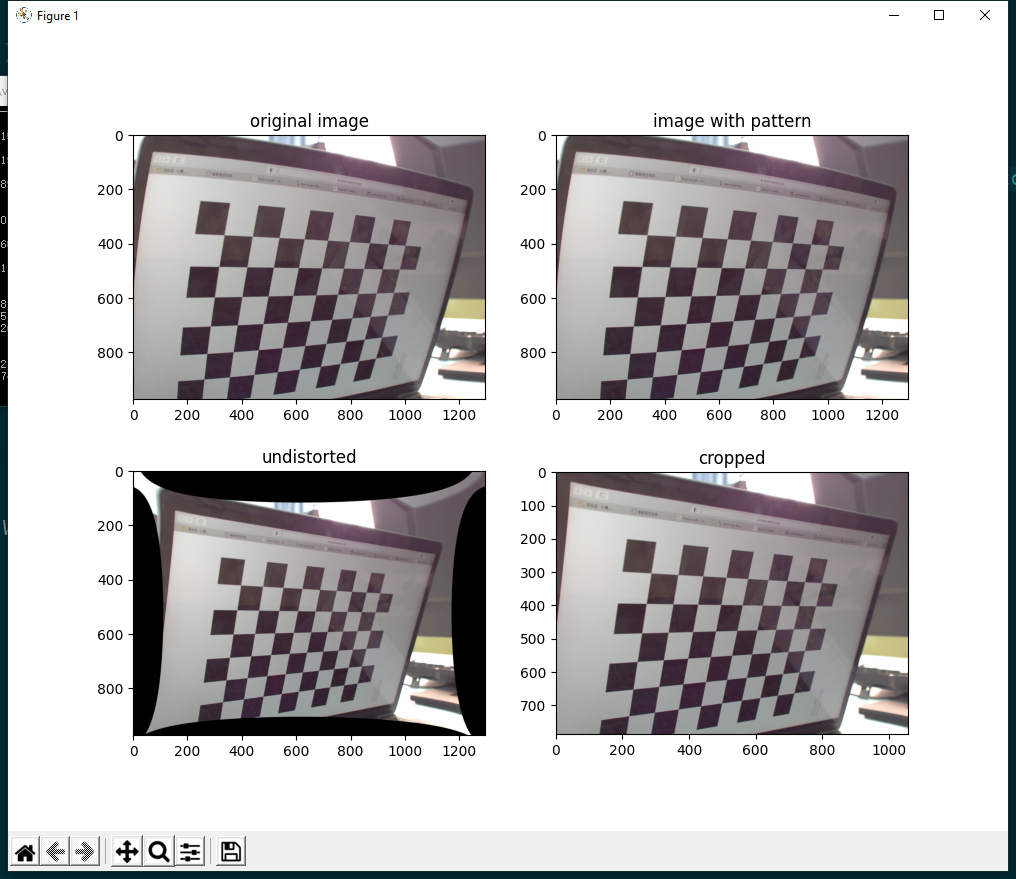
标定棋盘格：



校正后的图像：



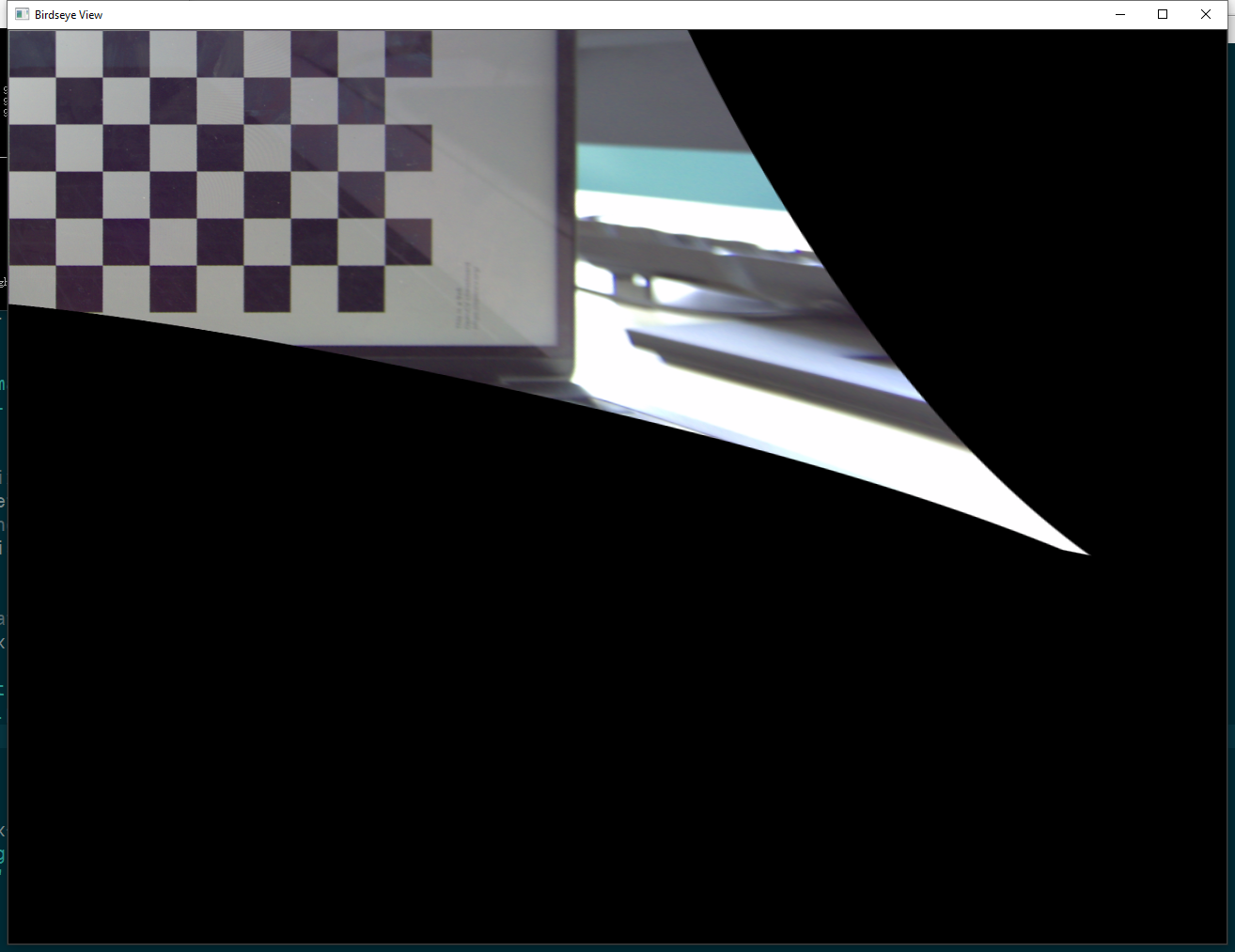
输出了整个过程中的图像：

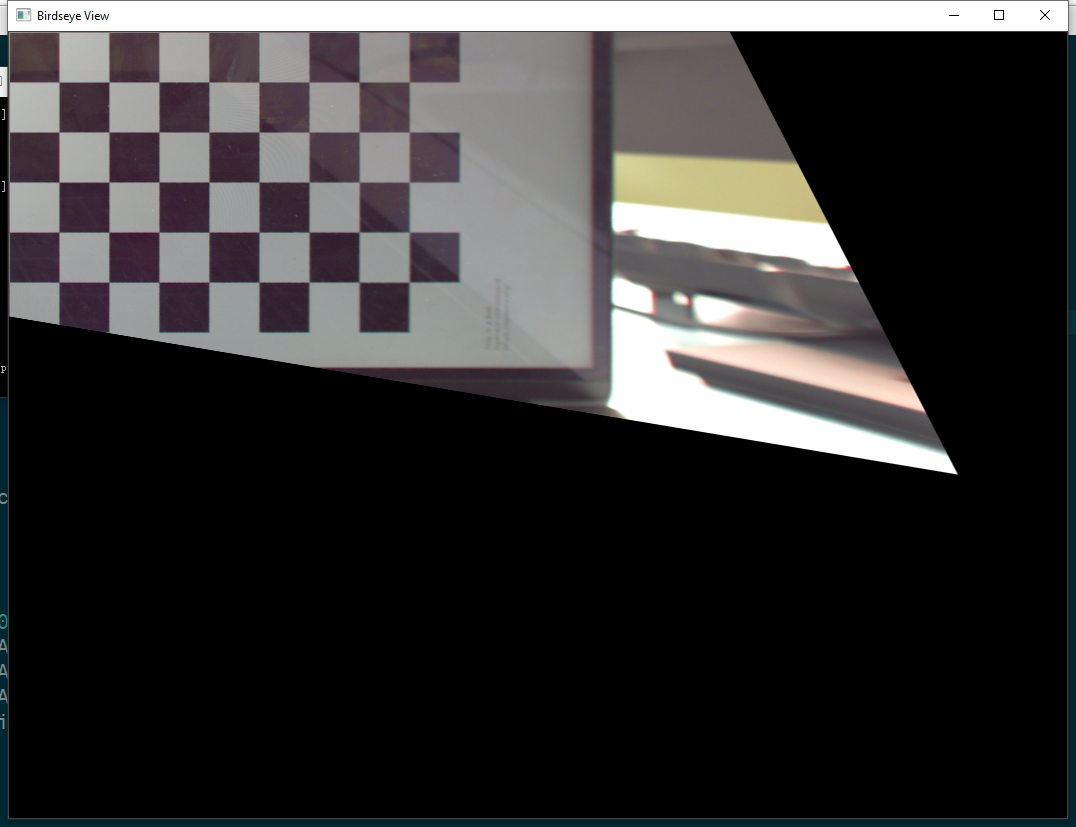


**输出的xml文件：**

|  |
| --- |
| <?xml version="1.0"?>  [<opencv\_storage><mtx type\_id="**opencv-matrix**">](file:///C:\Users\Flaze\Desktop\cv-lab-4\matrices.xml)<rows>3</rows><cols>3</cols>  <dt>d</dt>  <data> 1.9185538075266456e+03 0. 6.6351052686831258e+02 0. 1.9239847846858713e+03 5.2501768650563508e+02 0. 0. 1.</data></mtx>[<dist type\_id="**opencv-matrix**">](file:///C:\Users\Flaze\Desktop\cv-lab-4\matrices.xml)<rows>1</rows><cols>5</cols>  <dt>d</dt>  <data> -6.5131737583550275e-01 1.2672234354930185e+00 -7.6049220720545881e-03 -1.1437164634492115e-02 -5.9703308825363361e+00</data></dist>[<rvecs type\_id="**opencv-matrix**">](file:///C:\Users\Flaze\Desktop\cv-lab-4\matrices.xml)<rows>3</rows><cols>1</cols>  <dt>d</dt>  <data> 3.2420766902989018e-01 -6.4589768076974197e-01 1.1011907909749442e-01</data></rvecs>[<tvecs type\_id="**opencv-matrix**">](file:///C:\Users\Flaze\Desktop\cv-lab-4\matrices.xml)<rows>3</rows><cols>1</cols>  <dt>d</dt>  <data> -2.6876180482710543e+00 -1.2541719558169395e+00 1.5495572059324372e+01</data></tvecs></opencv\_storage> |

**鸟瞰视角**

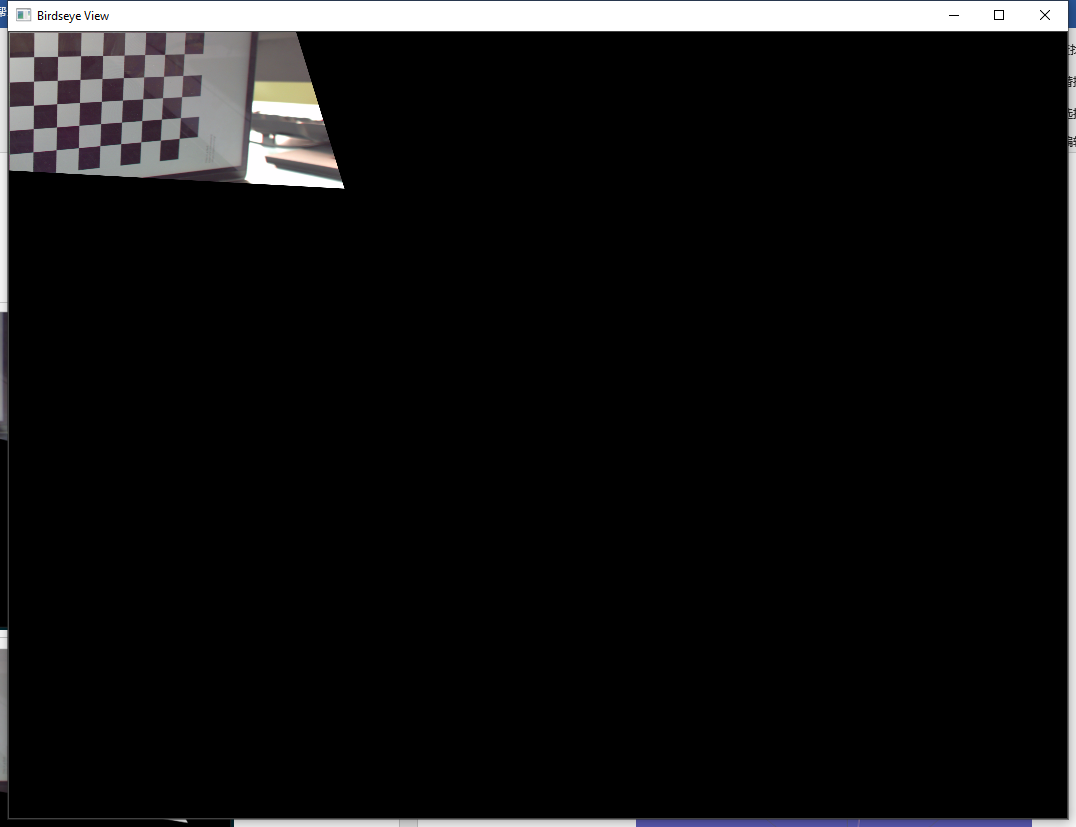




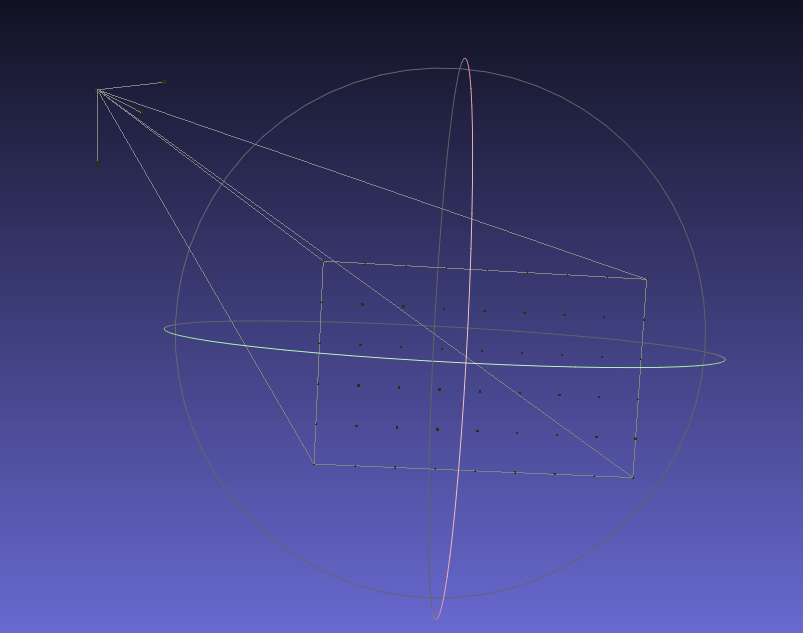
很明显，将棋盘格映射到了左上角，并且可以通过d和u来控制视角升高与降低

其中上一张图是使用了crop前的图像，在上升下降时会产生较严重的畸变。



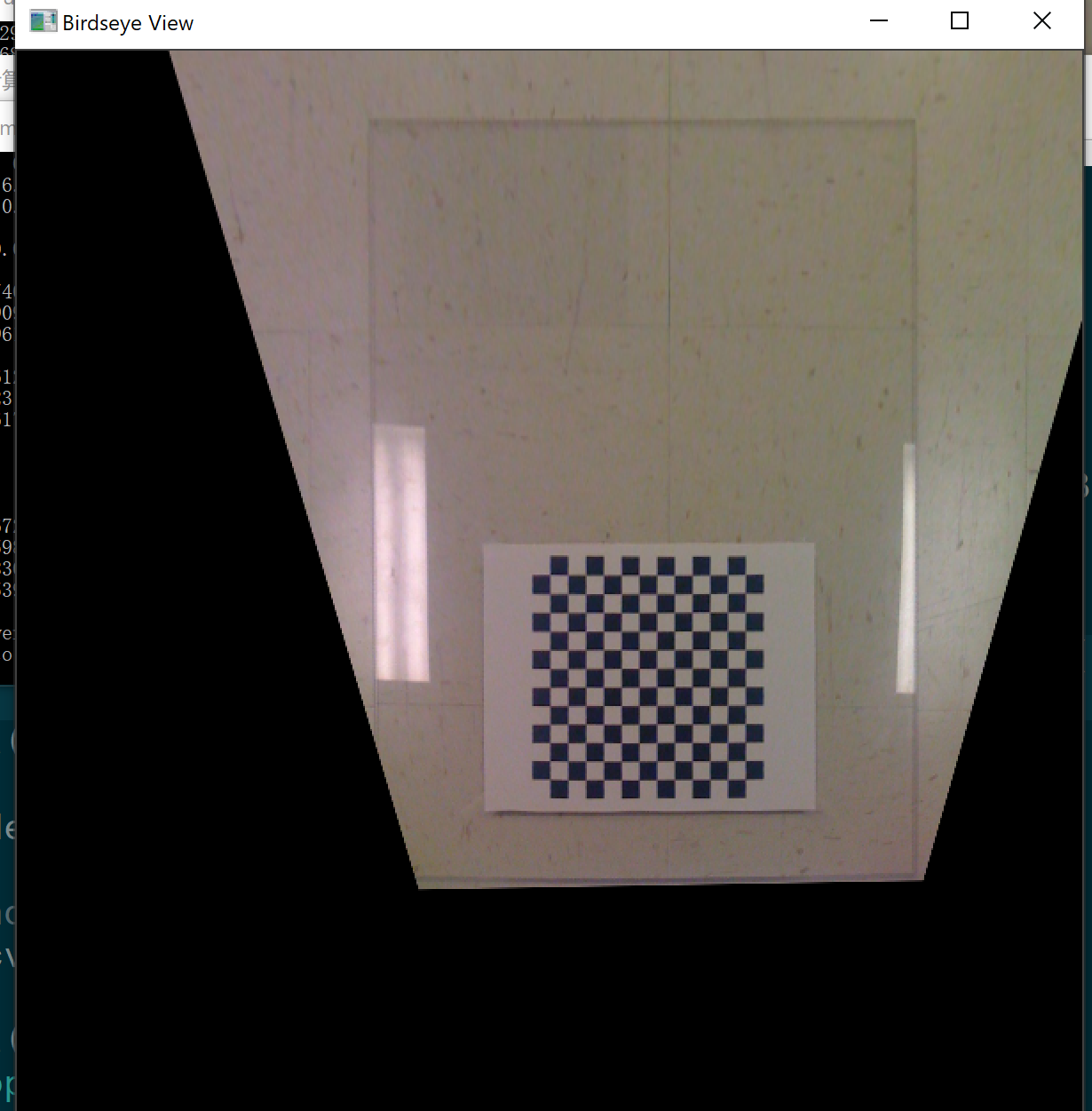


（顺便利用以前写的小工具生成了相机视角的ply文件）：



1. **课程给出的测试图片（由于效果非常好所以不做undistort直接运行）**

由于电脑内存较小，所以稍微改了一下对老师给出的图像进行处理的程序。希望验收的时候使用我自己拍的图（效果更好QAQ）



**五、心得体会**

在之前的智能视觉信息处理课程中，也使用棋盘格进行过图像的矫正，但当时只把这个当成基本工具，跑起来就好了（直接使用了官方给出的C++代码）。这次用python自己实现了一边，对它的操作流程更加熟悉了。

在做鸟瞰图的时候一开始一直生成非常模糊的图像，在查资料的过程中发现也有一个人（其实整个百度都是各个爬虫网站复制的这个提问x）出现这种问题，鼓捣鼓捣发现好像是生成视角变换矩阵时把src和dst弄反了，直接swap一下参数位置就好。

总感觉在有相机外参的时候，应该也可以利用相机坐标系和世界坐标系之间的关系来求视角转换，而不是使用像素点和棋盘格坐标的变换（但是本质应该是相同的），这也导致我在做实验时一直在考虑怎么对每一张图都做calibrate和视角转换。