**目标：**

本章节你需要学习以下内容:

* 我们将学习利用calib3d模块在图像中创建一些3D效果。

## api：

### numpy的savez方法保存文件以及用numpy.load方法来加载文件：

### savez(file, \*args, \*\*kwds):

***>>> from tempfile import TemporaryFile  
>>> outfile = TemporaryFile()  
>>> x = np.arange(10)  
>>> y = np.sin(x)  
  
Using `savez` with \\\*args, the arrays are saved with default names.  
  
>>> np.savez(outfile, x, y)  
>>> \_ = outfile.seek(0) # Only needed here to simulate closing & reopening file  
>>> npzfile = np.load(outfile)  
>>> npzfile.files  
['arr\_0', 'arr\_1']  
>>> npzfile['arr\_0']  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])  
  
Using `savez` with \\\*\*kwds, the arrays are saved with the keyword names.  
  
>>> outfile = TemporaryFile()  
>>> np.savez(outfile, x=x, y=y)  
>>> \_ = outfile.seek(0)  
>>> npzfile = np.load(outfile)  
>>> sorted(npzfile.files)  
['x', 'y']  
>>> npzfile['x']  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])***

### # Find the rotation and translation vectors.

### ret,rvecs, tvecs = cv2.solvePnP(objp, corners2, mtx, dist)

### # project 3D points to image plane

### imgpts, jac = cv.projectPoints(axis, rvecs, tvecs, mtx, dist)

**基础**

在上一节的摄像机标定中，我们已经得到了摄像机矩阵，畸变系数等。有了这些信息我们就可以估计图像中图案的姿态，或物体在空间中的位置，比如目标对象是如何摆放，如何旋转等。对一个平面对象来说，我们可以假设 Z=0，这样问题就转化成摄像机在空间中是如何摆放（然后拍摄）的。所以，如果我们知道对象在空间中的姿态，我们就可以在图像中绘制一些 2D 的线条来产生 3D 的效果。我们来看一下怎么做吧。

我们的问题是，在棋盘的第一个角点绘制 3D 坐标轴（X，Y，Z 轴）。X轴为蓝色，Y 轴为绿色，Z 轴为红色。在视觉效果上来看，Z 轴应该是垂直与棋盘平面的。

首先，让我们从先前的校准结果中加载相机矩阵和畸变系数。

**import numpy as np**

**import cv2 as cv**

**import glob**

**# Load previously saved data**

**with np.load('B.npz') as X:**

**mtx, dist, \_, \_ = [X[i] for i in ('mtx','dist','rvecs','tvecs')]**

现在让我们创建一个函数，绘制它获取棋盘中的角（使用cv.findChessboardCorners()获得）和轴点来绘制3D轴。

**def draw(img, corners, imgpts):**

**corner = tuple(corners[0].ravel())**

**img = cv.line(img, corner, tuple(imgpts[0].ravel()), (255,0,0), 5)**

**img = cv.line(img, corner, tuple(imgpts[1].ravel()), (0,255,0), 5)**

**img = cv.line(img, corner, tuple(imgpts[2].ravel()), (0,0,255), 5)**

**return img**

然后与前面的情况一样，我们创建终止标准，对象点（棋盘中的角点的3D点）和轴点。轴点是3D空间中用于绘制轴的点。我们绘制长度为3的轴（单位将以国际象棋方形尺寸表示，因为我们根据该尺寸校准）。所以我们的X轴是从（0,0,0）到（3,0,0）绘制的，同样。Y轴也一样。对于Z轴，它从（0,0,0）绘制到（0,0，-3）。负值表示它是朝着（垂直于）摄像机方向

**criteria = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)**

**objp = np.zeros((6\*7,3), np.float32)**

**objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)**

**axis = np.float32([[3,0,0], [0,3,0], [0,0,-3]]).reshape(-1,3)**

很通常一样我们需要加载图像。搜寻 7x6 的格子，如果发现，我们就把它优化到亚像素级。然后使用函数:cv2.solvePnPRansac() 来计算旋转和变换。但我们有了变换矩阵之后，我们就可以利用它们将这些坐标轴点映射到图像平面中去。简单来说，我们在图像平面上找到了与 3D 空间中的点（3,0,0）,(0,3,0),(0,0,3) 相对应的点。然后我们就可以使用我们的函数 draw() 从图像上的第一个角点开始绘制连接这些点的直线了。搞定！！！

**for fname in glob.glob('left\*.jpg'):**

**img = cv.imread(fname)**

**gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)**

**ret, corners = cv.findChessboardCorners(gray, (7,6),None)**

**if ret == True:**

**corners2 = cv.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)**

**# Find the rotation and translation vectors.**

**ret,rvecs, tvecs = cv.solvePnP(objp, corners2, mtx, dist)**

**# project 3D points to image plane**

**imgpts, jac = cv.projectPoints(axis, rvecs, tvecs, mtx, dist)**

**img = draw(img,corners2,imgpts)**

**cv.imshow('img',img)**

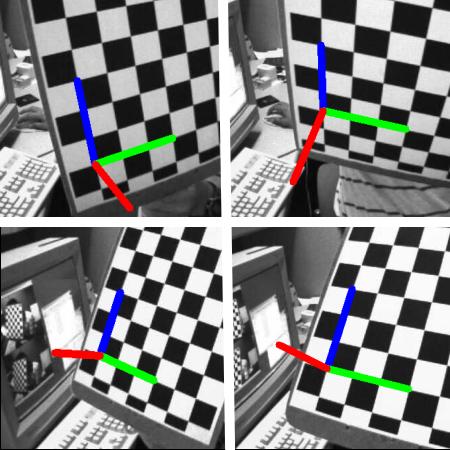
**k = cv.waitKey(0) & 0xFF**

**if k == ord('s'):**

**cv.imwrite(fname[:6]+'.png', img)**

**cv.destroyAllWindows()**

看下面的一些结果。请注意，每个轴的长度为3个方格：

[](https://camo.githubusercontent.com/788e1b84204221b49bb20d1c9c80eccafd612037bf348dbc851d3f521fe1d2fb/68747470733a2f2f646f63732e6f70656e63762e6f72672f342e302e302f706f73655f312e6a7067)

如果要绘制立方体，请按如下方式修改draw()函数和轴点。

修改了draw()函数：

**def draw(img, corners, imgpts):**

**imgpts = np.int32(imgpts).reshape(-1,2)**

**# draw ground floor in green**

**img = cv.drawContours(img, [imgpts[:4]],-1,(0,255,0),-3)**

**# draw pillars in blue color**

**for i,j in zip(range(4),range(4,8)):**

**img = cv.line(img, tuple(imgpts[i]), tuple(imgpts[j]),(255),3)**

**# draw top layer in red color**

**img = cv.drawContours(img, [imgpts[4:]],-1,(0,0,255),3)**

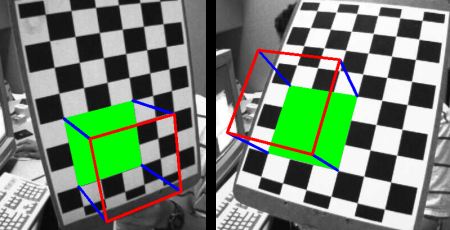
**return img**

修改了轴点。它们是3D空间中立方体的8个角：

**axis = np.float32([[0,0,0], [0,3,0], [3,3,0], [3,0,0],**

**[0,0,-3],[0,3,-3],[3,3,-3],[3,0,-3] ])**

结果如下图所示：

[](https://camo.githubusercontent.com/b3373998b1d5c5a6c217e4d52e50c259f872d79d99e35d52f29a11c7211856fd/68747470733a2f2f646f63732e6f70656e63762e6f72672f342e302e302f706f73655f322e6a7067)

# 本节完整实例代码，注意这一节的代码依赖上一节生成的b2.npz文件，我们把它拷贝过来了。

## 实例1：post-estimation1.py

|  |  |
| --- | --- |
|  | **import cv2 import numpy as np import glob  def draw(img, corners, imgpts):  corner = tuple(corners[0].ravel())  img = cv2.line(img, corner, tuple(imgpts[0].ravel()), (255,0,0), 5)  img = cv2.line(img, corner, tuple(imgpts[1].ravel()), (0,255,0), 5)  img = cv2.line(img, corner, tuple(imgpts[2].ravel()), (0,0,255), 5)  return img   *# Load previously saved data* with np.load('b2.npz') as X:  mtx, dist, \_, \_ = [X[i] for i in ('mtx','dist','rvecs','tvecs')]  criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001) objp = np.zeros((6\*7,3), np.float32) objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)  axis = np.float32([[3,0,0], [0,3,0], [0,0,-3]]).reshape(-1,3)  for fname in glob.glob('left\*.jpg'):  img = cv2.imread(fname)  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  ret, corners = cv2.findChessboardCorners(gray, (7, 6), None)   if ret == True:  corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)   *# Find the rotation and translation vectors.* ret, rvecs, tvecs = cv2.solvePnP(objp, corners2, mtx, dist)   *# project 3D points to image plane* imgpts, jac = cv2.projectPoints(axis, rvecs, tvecs, mtx, dist)  img = draw(img, corners2, imgpts)  cv2.imshow('img', img)  k = cv2.waitKey(0) & 0xFF  if k == ord('s'):  cv2.imwrite(fname[:6] + '.png', img)  cv2.destroyAllWindows()** |

### 效果：会运行多次，效果大致如下

|  |
| --- |
| image4 |

## 实例2：post-estimation2-cube.py

|  |  |
| --- | --- |
|  | **import cv2 import numpy as np import glob   def draw(img, corners, imgpts):  imgpts = np.int32(imgpts).reshape(-1, 2)   *# draw ground floor in green* img = cv2.drawContours(img, [imgpts[:4]], -1, (0, 255, 0), -3)   *# draw pillars in blue color* for i, j in zip(range(4), range(4, 8)):  img = cv2.line(img, tuple(imgpts[i]), tuple(imgpts[j]), (255), 3)   *# draw top layer in red color* img = cv2.drawContours(img, [imgpts[4:]], -1, (0, 0, 255), 3)   return img   *# Load previously saved data* with np.load('b2.npz') as X:  mtx, dist, \_, \_ = [X[i] for i in ('mtx', 'dist', 'rvecs', 'tvecs')]  criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001) objp = np.zeros((6 \* 7, 3), np.float32) objp[:, :2] = np.mgrid[0:7, 0:6].T.reshape(-1, 2)  axis = np.float32([[0, 0, 0], [0, 3, 0], [3, 3, 0], [3, 0, 0],  [0, 0, -3], [0, 3, -3], [3, 3, -3], [3, 0, -3]])  for fname in glob.glob('left\*.jpg'):  img = cv2.imread(fname)  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  ret, corners = cv2.findChessboardCorners(gray, (7, 6), None)   if ret == True:  corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)   *# Find the rotation and translation vectors.* ret, rvecs, tvecs = cv2.solvePnP(objp, corners2, mtx, dist)   *# project 3D points to image plane* imgpts, jac = cv2.projectPoints(axis, rvecs, tvecs, mtx, dist)  img = draw(img, corners2, imgpts)  cv2.imshow('img', img)  k = cv2.waitKey(0) & 0xFF  if k == ord('s'):  cv2.imwrite(fname[:6] + '.png', img)  cv2.destroyAllWindows()** |

### 效果：会运行多次，效果大致如下

|  |
| --- |
| image5 |