# 计算机视觉实践-练习1

**代码实现说明：**

Stitch代码中，主要步骤如下：

特征匹配：使用features.generateHomography(src\_img, dst\_img)函数生成单应性矩阵H，该函数内部可能使用了特征检测和特征匹配算法来找到源图像和目标图像之间的对应关系。

图像配准：使用cv2.warpPerspective函数和单应性矩阵H将源图像变换到目标图像的视角，得到配准后的源图像。

图像融合：使用panoramaBlending函数将配准后的源图像和目标图像进行融合。这个函数使用了一个平滑窗口来生成一个过渡区域，使得源图像和目标图像在这个区域内平滑过渡，以减少拼接后的接缝。

多图像拼接：multiStitching函数实现了多图像的拼接。它首先选择中间的图像，然后将图像列表分为左右两部分，分别与中间图像进行拼接。

图像裁剪：最后，crop函数根据目标图像的大小对全景图像进行裁剪，去除黑色的无效区域。

Feature.py代码中，主要步骤如下：

特征检测和描述：findAndDescribeFeatures函数用于在图像中找到并描述特征。它首先将图像转换为灰度图像，然后根据输入的选项（“SURF”，“ORB"或"SIFT”）选择相应的特征检测和描述算法。这些算法都是用于在图像中检测和描述局部特征的方法。函数返回图像的关键点和特征。

特征匹配：matchFeatures函数用于匹配两个特征集。它可以选择使用"BF"（BruteForce）或"FB"（FlannBased）算法。这两种算法都是用于在特征向量集之间进行匹配的方法。函数返回匹配的特征。

生成单应性矩阵：generateHomography函数用于生成源图像和目标图像之间的单应性矩阵。它首先找到并描述两个图像的特征，然后匹配这些特征，最后使用RANSAC算法找到最佳的单应性矩阵。

绘制关键点和匹配：drawKeypoints和drawMatches函数用于在图像上绘制关键点和匹配的特征。这可以帮助我们可视化特征检测和匹配的结果。

Utils代码中，主要步骤如下：加载图像：loadImages函数从指定路径加载图像，并将其转换为数组。如果resize参数为1，那么图像的大小将被缩小到原来的四分之一。

裁剪图像：trim函数用于裁剪图像。它从图像的四个边缘开始，如果某个边缘的所有像素值都为0（即该边缘是黑色的），那么就裁剪掉这个边缘。这个过程会一直重复，直到所有的边缘都不是黑色为止。

给图像添加边框：padding函数用于给图像添加边框。它使用cv2.copyMakeBorder函数来添加边框，可以指定上、下、左、右四个方向的边框宽度。这个函数可以用于在图像的边缘添加空白区域，以防止在后续的图像处理过程中边缘信息的丢失

**运行截图说明：**

这个函数用于将OpenCV加载的图像转换为适合使用Matplotlib显示的格式。首先，它将图像的数值范围从[0, 255]缩放到[0, 1]，以确保图像的值在正确的范围内。然后，它将图像从BGR格式转换为RGB格式，因为Matplotlib默认使用RGB格式显示图像。

def convertResult(img):

img = np.array(img,dtype=float)/float(255)

img = img[:,:,::-1]

return img

加载图像

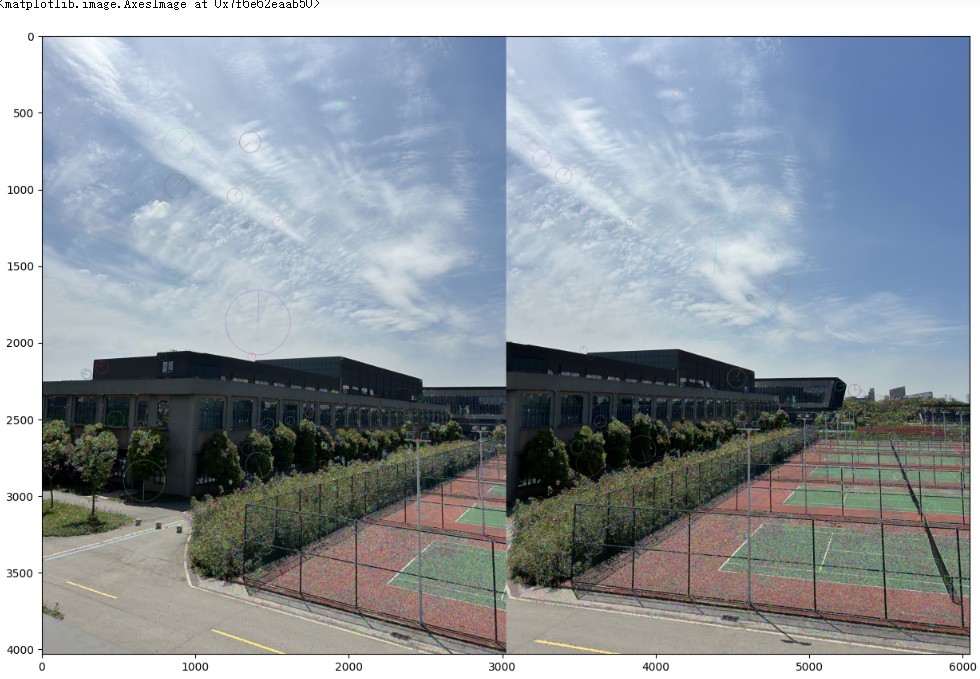
list\_images=utils.loadImages('data/nlg',resize=0)

使用SIFT算法提取图像的关键点和描述符

k0,f0=features.findAndDescribeFeatures(list\_images[0],opt='SIFT')

k1,f1=features.findAndDescribeFeatures(list\_images[1],opt='SIFT')

将提取到的关键点在原始图像上绘制出来，并将两幅图像的关键点拼接在一起进行显示



在两幅图像的特征描述之间进行匹配

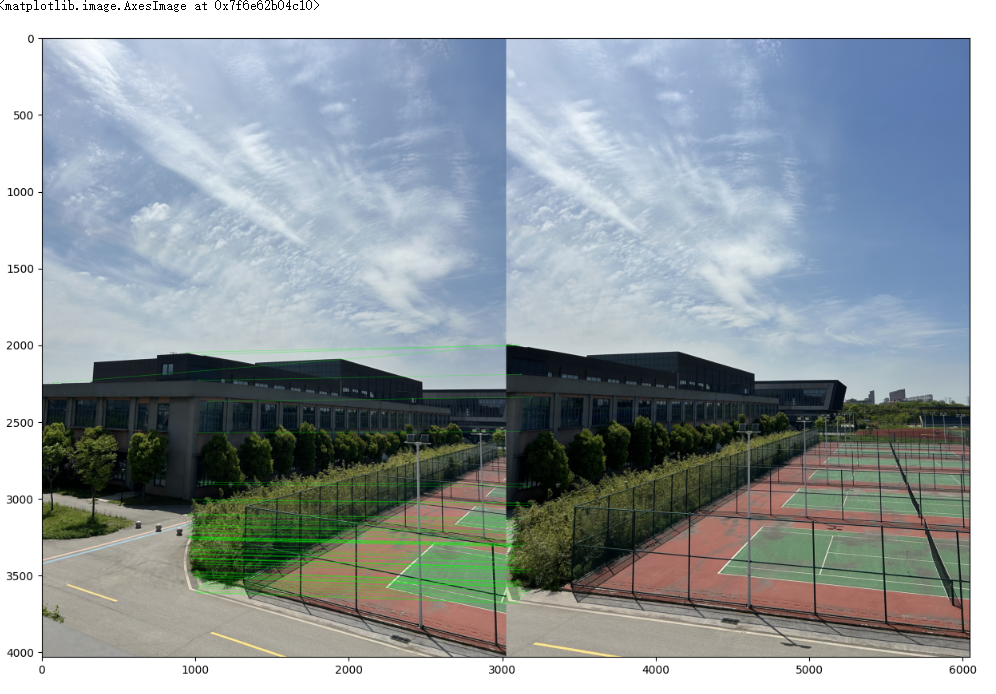
mat=features.matchFeatures(f0,f1,ratio=0.6,opt='BF')

计算两幅图像之间的单应性矩阵

H,matMask=features.generateHomography(list\_images[0],list\_images[1])

用于在两幅图像上绘制匹配的特征点，并使用单应性矩阵和匹配掩码来筛选有效的匹配

img=features.drawMatches(list\_images[0],k0,list\_images[1],k1,mat,matMask) plt.figure(figsize=(15,15)) plt.imshow(convertResult(img))



将两幅图像进行拼接和融合形成全景图像

pano,non\_blend,left\_side,right\_side=stitch.warpTwoImages(list\_images[1],list\_images[0],True)

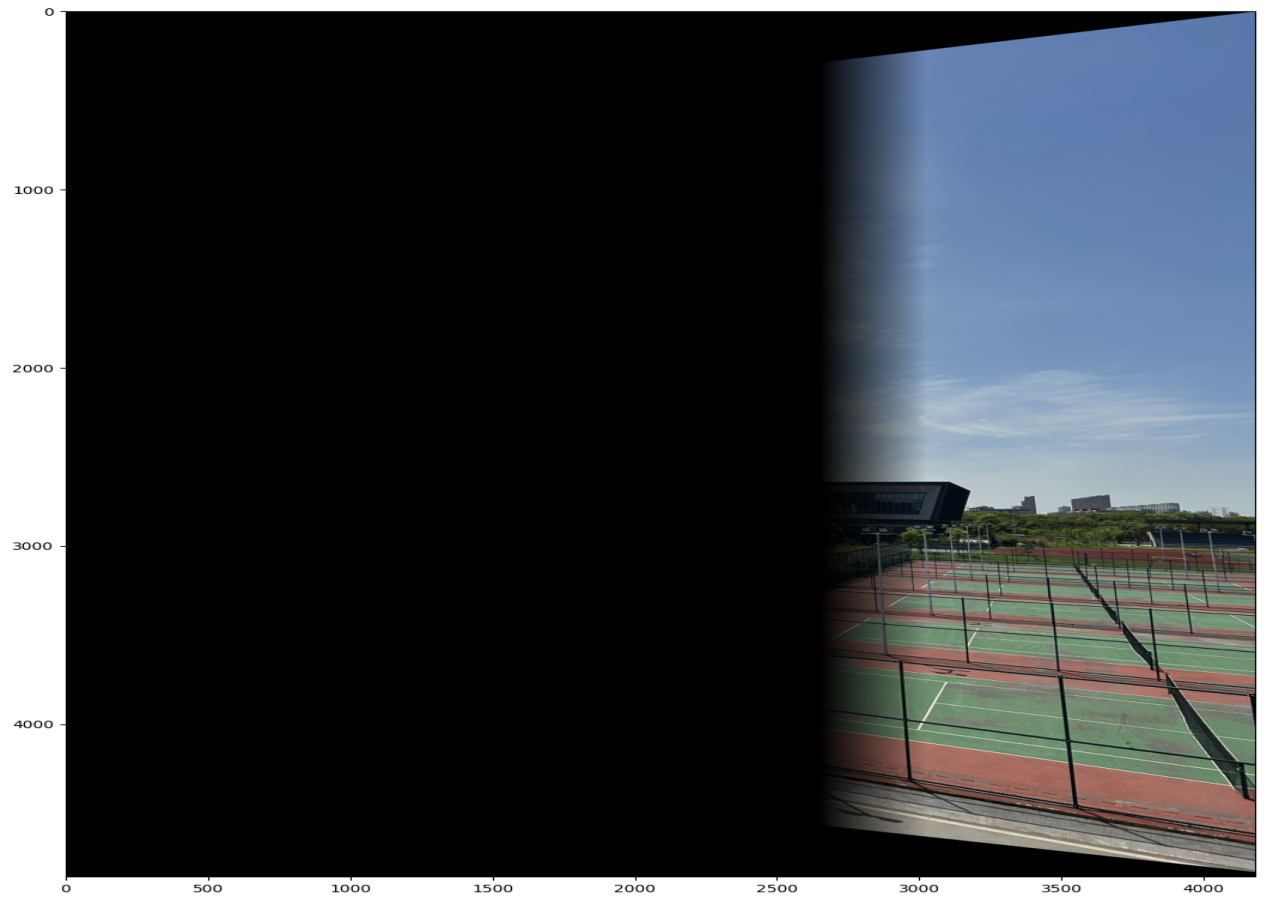
plt.figure(figsize=(15,15))

plt.imshow(convertResult(left\_side))



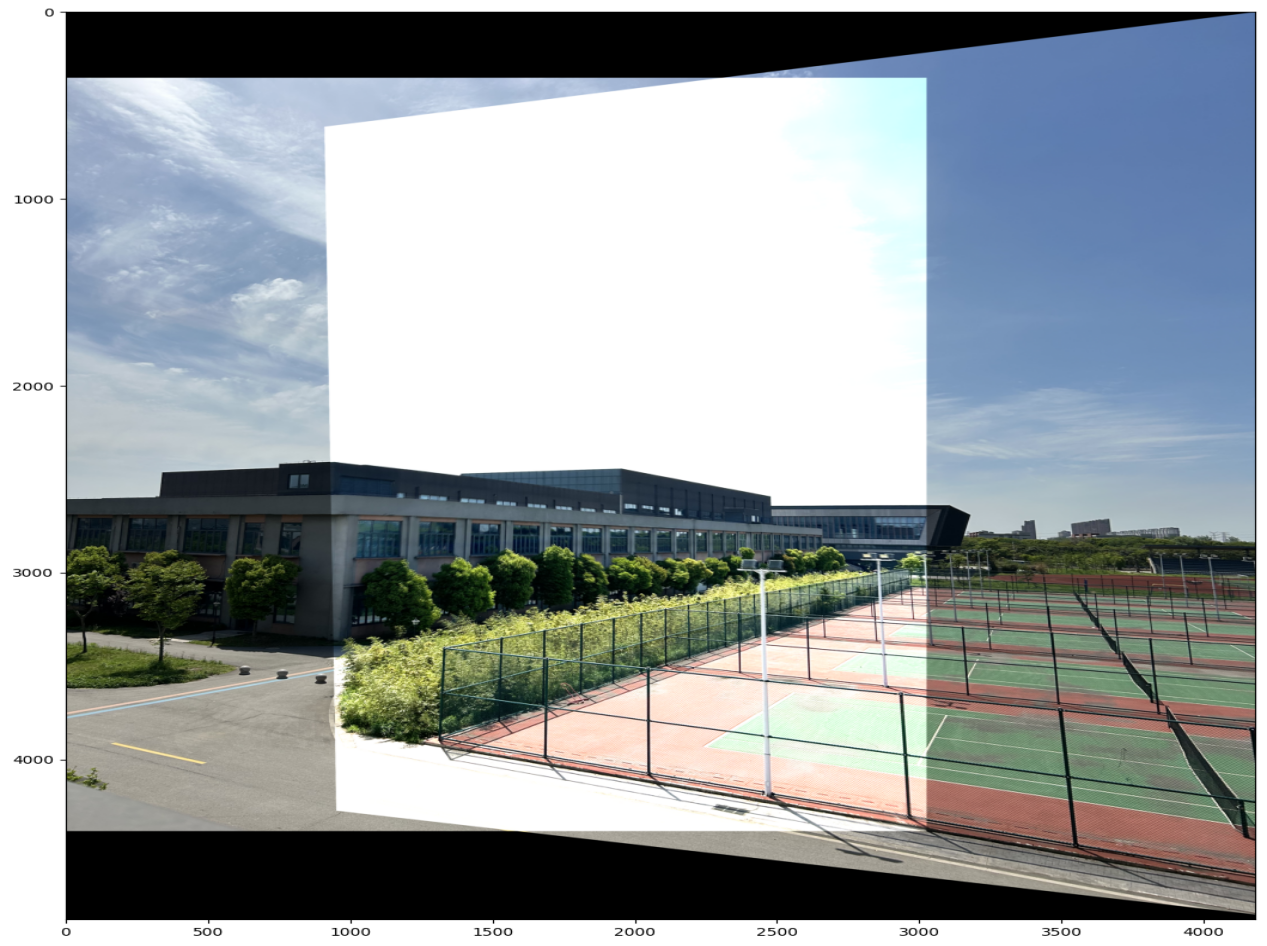
plt.figure(figsize=(15,15))

plt.imshow(convertResult(right\_side))



plt.figure(figsize=(15,15))

plt.imshow(convertResult(non\_blend))



\_,non\_blend2,\_,\_=stitch.warpTwoImages(list\_images[0],list\_images[1],True)

plt.figure(figsize=(15,15))

plt.imshow(convertResult(non\_blend2))



plt.figure(figsize=(15,15))

plt.imshow(convertResult(pano))



import utils

list\_images=utils.loadImages('data/nlg',resize=0)

panorama=stitch.multiStitching(list\_images)

plt.figure(figsize=(20,20))

plt.imshow(convertResult(panorama))

|  |  |
| --- | --- |
| 001 | 002 |
| 003 | 004 |

最终结果：

