

# 计算机视觉第六次作业

## 作业要求

1. Take pictures on a tripod (or handheld)
2. Warp images to spherical coordinates
3. Extract SIFT features and Match features (by KNN or Hashing)
4. Align neighboring pairs using RANSAC
5. Write out list of neighboring translations
6. Correct for drift
7. Read in warped images and blend them (using multi-scale blending or Poisson blending)
8. Crop the result and import into a viewer

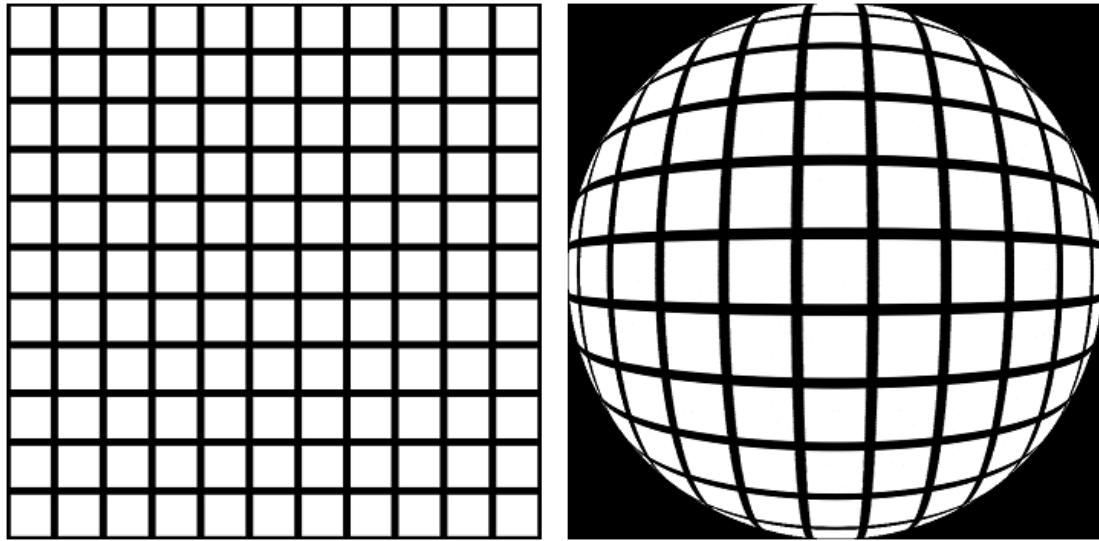
## 分析简述

作业要求对输入的图片进行拼接生成全景的图像，主要的几个步骤为首先将图片转换至球面坐标，然后使用 SIFT 对各图的特征点进行识别，再利用 kd 树进行特征的匹配找出相邻图像。对于相邻的图像，使用 RANSAC 算法筛选匹配的特征点并计算变换矩阵，最后利用拉普拉斯金字塔融合进行变换后的图像的拼接。

对于流程各个部分的简述如下：

## 球面变换

球形变换的过程大致是将下图中的左图变换至右图的形态：



变换的方法的大致过程是将图像中心点作为球心  $O$ , 半径为  $r$ , 对于每一个球上的点  $(x_r, y_r)$ , 用如下公式算出其在原图像的坐标  $(x_o, y_o)$ .

$$\text{弧长 } OA = \frac{\arcsin\left(\frac{\sqrt{x_r^2 + y_r^2}}{r}\right) * 2r}{\pi}$$

$$x_o = OA \times \cos(\arctan \frac{y_r}{x_r})$$

$$y_o = OA \times \sin(\arctan \frac{y_r}{x_r})$$

然而，在提交的实际的程序中，因为经测试发现球面变化后的图像进行特征点匹配时达不到预期的效果，因此实际使用的是柱面变化进行操作。

## SIFT

SIFT 全名为尺度不变特征转换，用于检测在不同尺度空间的图像下的具有方向信息的局部极值点，其算法主要过程为：生成尺度空间，检测尺度空间极值点，精确定位极值点，为每个关键点指定方向参数，生成关键点描述子。

尺度空间的生成过程为使用高斯平滑和降采样构建高斯金字塔，计算 DOG 算子，然后使用算子找出比其邻域 8 个点以及其上下相邻尺度的邻域 9 个点共 26 个点均更大的极值点作为 DOG 函数的局部极值点。对于求出的局部极值点，丢弃对比度低的特征点并使用 Hessian 矩阵去除边缘响应。采用梯度直方图统计法进行关键点方向参数计算，最后生成具有独特性的向量构建 128 维关键点描述子。

在提交的代码中使用了 VLfeat 库进行 SIFT 的计算以及特征点的匹配。

## RANSAC

RANSAC 全称为随机抽样一致算法，用于估算出数学模型中的参数。与常用的最小二乘法相比，RANSAC 可以减小离群数据对结果的影响。其主要流程为：在所有数据中随机寻找一定数量的点作为内群，从内群中拟合出对应的数学模型，然后将其他点代入模型记录内群数量，重复迭代一定次数后选出内群数量最多的一组，使用组中所有内群重新拟合获得最终的结果。其迭代次数的计算方法如下：

$$k = \frac{\log(1 - P)}{\log(1 - p^n)}$$

其中  $P$  为经过所有迭代后找到正确结果的概率， $p$  为数据中内群的比例， $n$  为每次随机选取作为内群的数据数量。

## 单应性矩阵

RANSAC 的过程中每一次进行拟合的过程为使用最小二乘法计算单应性矩阵，单应性指一个平面到另一个平面的投影映射。单应性矩阵是一个  $3*3$  的矩阵，有着 8 个自由度。

## 拉普拉斯金字塔融合

提交的代码中使用了拉普拉斯金字塔融合进行邻接图像的拼接。在先前 SIFT 中提到的高斯金字塔基础之上，用每一层图像减去上一层图像上采样并高斯卷积后的预测图像，得到的差值图像即为拉普拉斯金字塔图像。分别求得要融合的图像的拉普拉斯金字塔并拼起来后，从上到下差值并于下层相加，获得最终结果。

## 代码结构简述

提交的代码总共有 19 个代码文件，8 个类，分别介绍如下：

1. utils.h 和 utils.cpp 文件包含了双线性差值和 RGB 图转灰度图函数的定义与实现。
2. spherical\_warp.h 和 spherical\_warp.cpp 包含了 spherical\_warp 类的定义与实现，由于结果效果不佳，在实际的拼接中没有调用这个类
3. cylindrical\_warp.h 和 cylindrical\_warp.cpp 文件包含了 cylindrical\_warp 类的定义与实现，在对图像进行 SIFT 特征点识别前调用此类进行图像柱面转换。
4. feature\_extraction.h 和 feature\_extraction.cpp 包含了 feature\_extraction 类，此类调用了 VLFeat 库进行 SIFT 特征点提取。
5. feature\_matching.h 和 feature\_matching.cpp 包含 feature\_matching 类，调用 VLFeat 库对上类获得的特征点进行匹配，返回匹配的特征点对。
6. neighboring\_translation.h 和 neighboring\_translation.cpp 包含的 neighboring\_translation 类对重复调用 feature\_matching 类获得的邻接图像的表，获得拼接图像时的图像序列。

7. RANSAC.h 和 RANSAC.cpp 包含了 RANSAC 类，对输入的特征点对计算单应性矩阵
8. blending.h 和 blending.cpp 包含 blending 类，类中包含了对输入图像根据单应性矩阵进行变换的方法以及通过拉普拉斯金字塔融合拼接图像的方法。
9. image\_stitching.cpp 和 image\_stitching.h 包含 image\_stitching 类，类中 run\_stitching 方法包含了拼接的大部分过程的函数调用，返回完成拼接后的图像。
10. main.cpp 是程序的主函数，运行时从命令行接收两个参数，第一个参数为图片数目，第二个参数为图片的目录，运行过程会从目录下接 0.jpg, 1.jpg, … 的顺序读至指定的图片数目。对于尺寸较大的图片，将会进行 resize 到一定尺寸后构建 image\_stitching 类并调用 run\_stitching 方法执行拼接，拼接完成后将显示出获得的图像并将图像保存到文件夹下的 panorama.jpg 文件。

## 程序编译

程序的运行需要依赖 VLFeat 库，在编译前要设置环境变量 VLROOT 到 VLFeat 库的根目录下。然后使用 cmake 进行编译，编译命令如下：

```
# Linux/Mac
cmake .
make

#Windows
cmake -G "MinGW Makefiles" .
mingw32-make
```

编译后 make 会将可执行文件输出至 bin 文件夹下并将代码目录下的对应平

台的动态链接库文件拷贝至 bin 目录下，Mac 和 Windows 可以直接移至 bin 目录下运行，Linux 系统需要修改/etc/ld.so.conf 文件将动态链接库存放的目录添加至文件中才可运行。提交的压缩包中已在 code/bin 文件夹下包含了三个平台的可执行文件和动态链接库。

## 程序执行效果

1. 球形变换效果：由于在最终程序中没有实际用到球形变换，此处仅展示执行变换后的效果：



实际使用的柱形变换的效果如下：



2. 对给出的 TEST-ImageData(1)的执行效果：

```
./panorama 4 "../TEST-ImageData(1)"
```



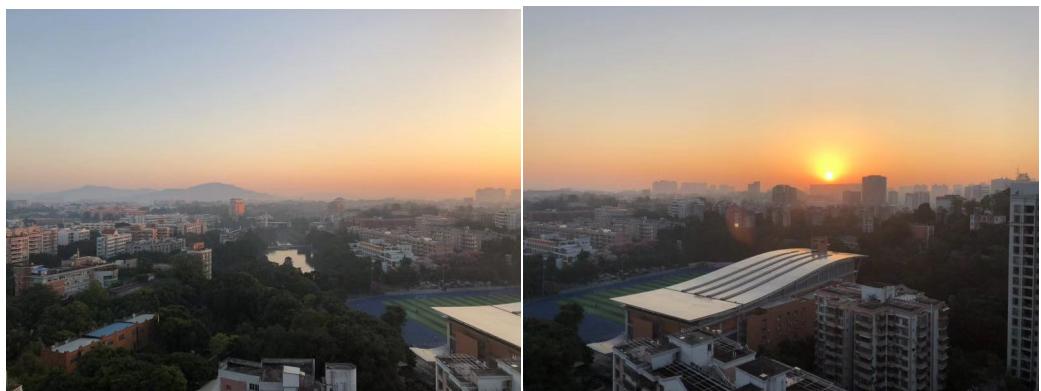
3. 对给出的 TEST-ImageData(2)的执行效果：

```
./panorama 18 "../TEST-ImageData(2)"
```

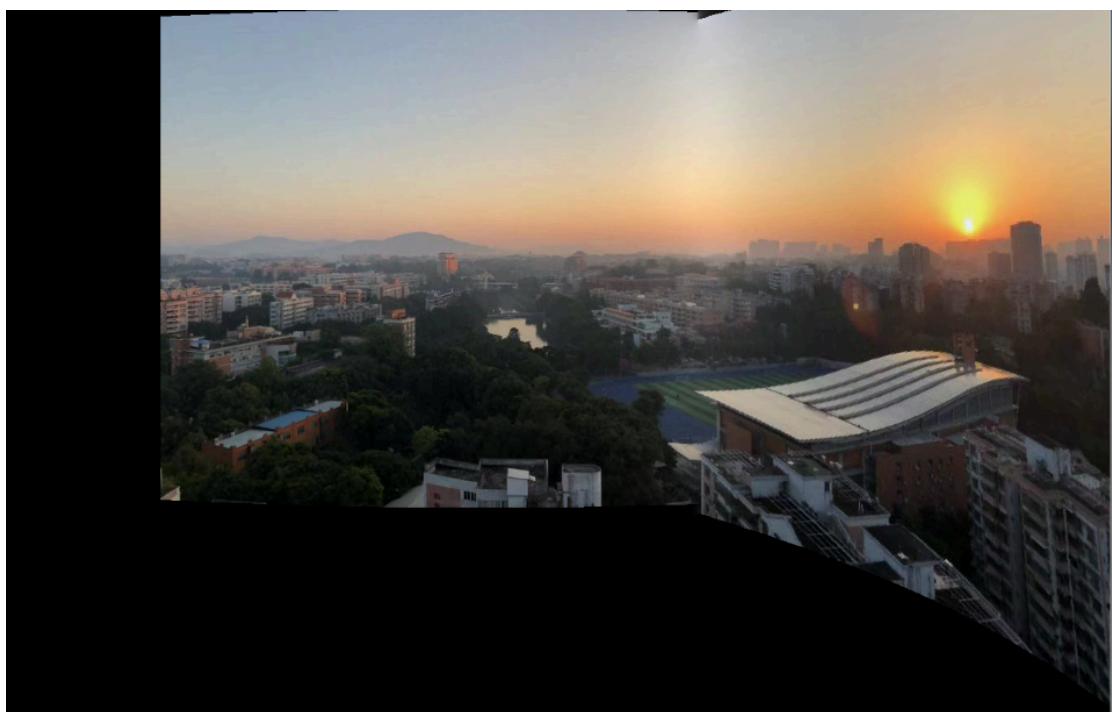


4. 原图如下（在华工拍摄）：

```
./panorama 2 ../TEST8
```



拼接后效果如下：

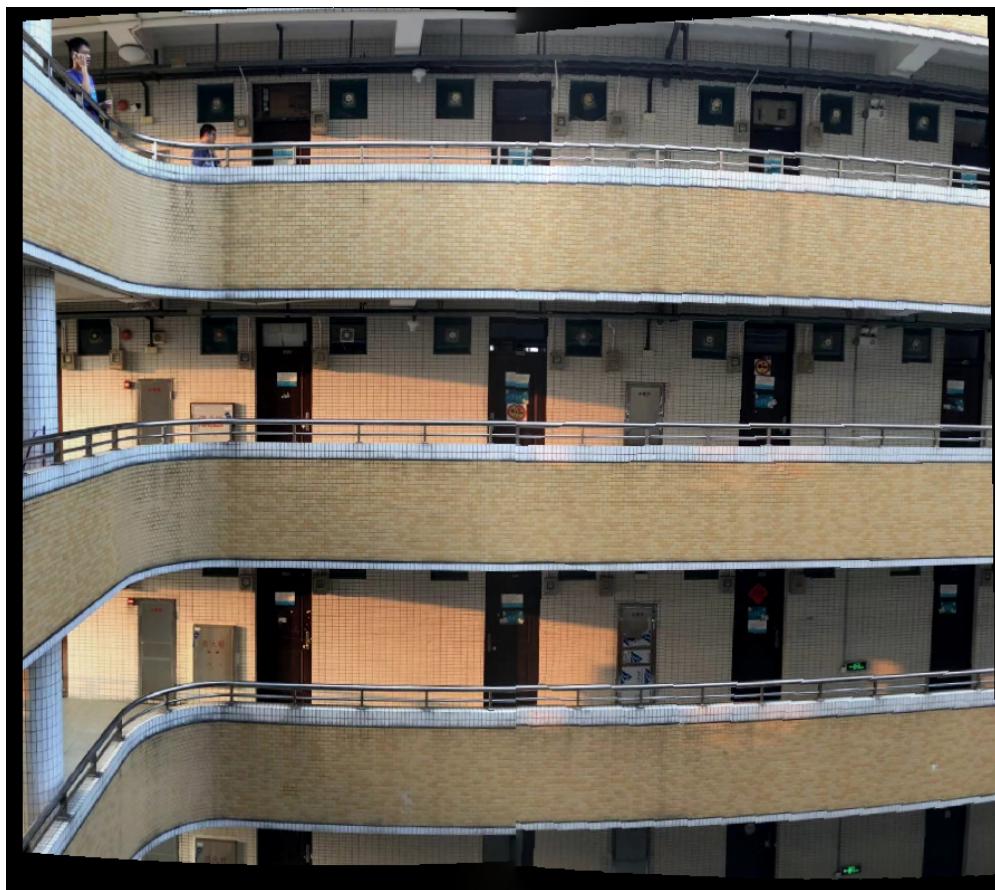


5. 原图如下（在宿舍区拍摄）：

./panorama 2 ../../TEST4



拼接后效果如下：

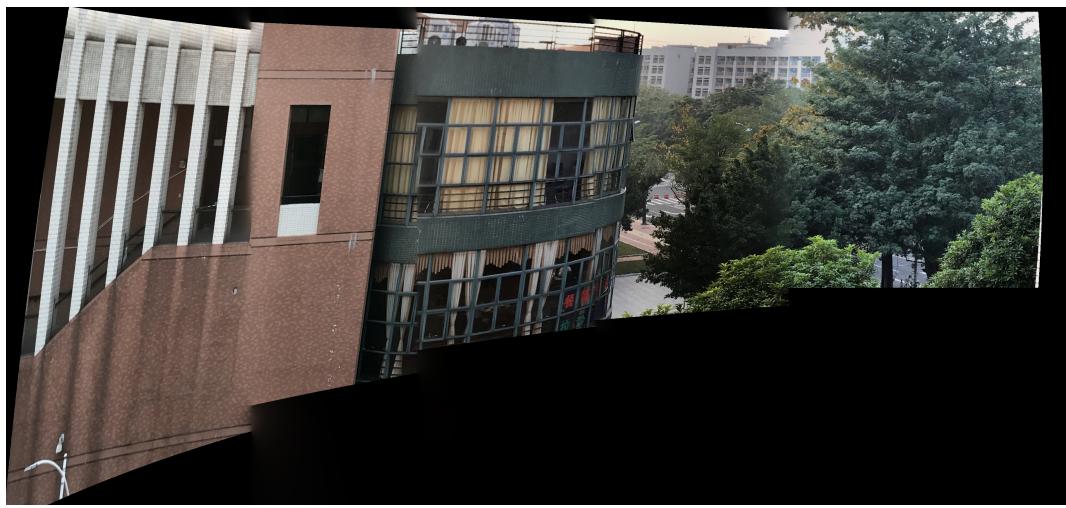


6. 原图如下（饭堂拍摄）：

./panorama 5 ../../TEST5



拼接后效果如图：



## 结果分析

由上面几个样例可看出程序仍然存在部分问题：在要拼接的图片较多时效果较差，当图片中物体离镜头较近时结果也会出现非预期的效果，对于相邻图片的连接处处理仍然存在不自然和不连续的情况，图片间的色差也较为明显。在处理过程中也偶尔会有程序异常退出的问题。