**EX6—图像拼接—实验报告**

1. 测试环境

Windows10 64bits，mingw

g++ -o main main.cpp –O2 –lgdi32

编译直接输入make即可（一个bat文件）

运行为main

Matlab版本直接运行test即可

1. 测试数据

在./input\_s文件夹里，共18张图片，我进行了重命名，其中前四张图片对应dataset1里的4张图片，全部18张图片对应dataset2里的18张图片。

1. 实验代码编写和基本算法

本次作业要求利用sift特征和ransac（random sample consensus）算法实现图像的拼接，算法分为以下几步，我实现了C++版本（完全版）和Matlab（不完全版）版本，下面详细说明：

1. 对图像做一个warping的预处理：

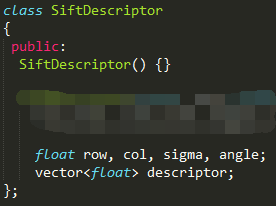
由于图像是用某种镜头拍摄的，拍出来的图片并不在plain space上，故需要做一个柱面投影消除这种镜头带来的影响。代码如下



做完柱面投影后，图像被映射回plain space上，但是图像上下左右会有一些黑边，这些黑边在blending那步会提及和解决。

1. 检测待拼接的两幅图像的SIFT特征：

由于sift特征的检测比较复杂，故没有自己实现而是调用了一个库进行检测，检测结果为一个自定义类型如下：

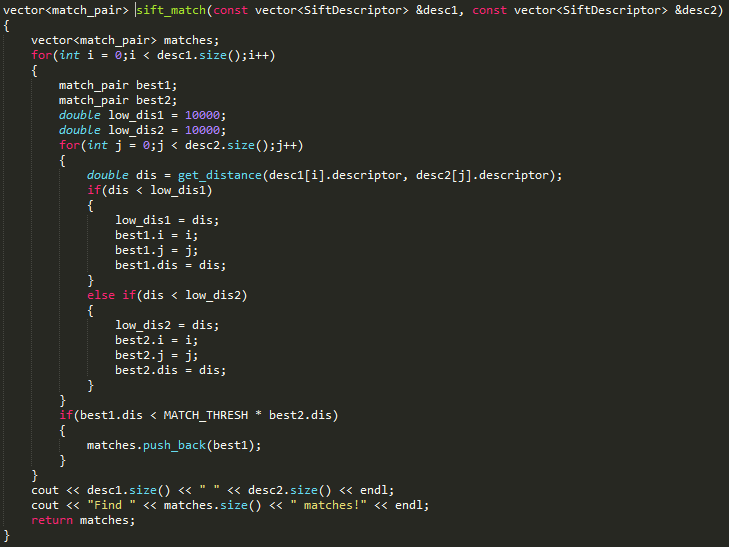


变量名顾名思义，其中description为一个128维向量，记录了特征点的特征直方图。Sift算法我的理解如下：

对图像做高斯模糊和下采样，得到不同尺度不同模糊度的高斯金字塔，然后对金字塔里相邻的高斯平滑图像进行相减得到DOG即高斯差分金字塔。然后对DOG里的点找极值点（三维空间上，与周围3\*3\*3-1=26个点比较得出）。然后，为每个关键点确定其方向参数。至此特征点属性有：位置（row、column），所处尺度，方向（通过统计该点附近的点对该关键点的方向生成的贡献，得出一个梯度直方图，然后取峰值）。最后对关键点周围八个方向的区域计算梯度直方图，得出一个4\*4的向量，共8个方向故描述子的维度为4\*4\*8=128.

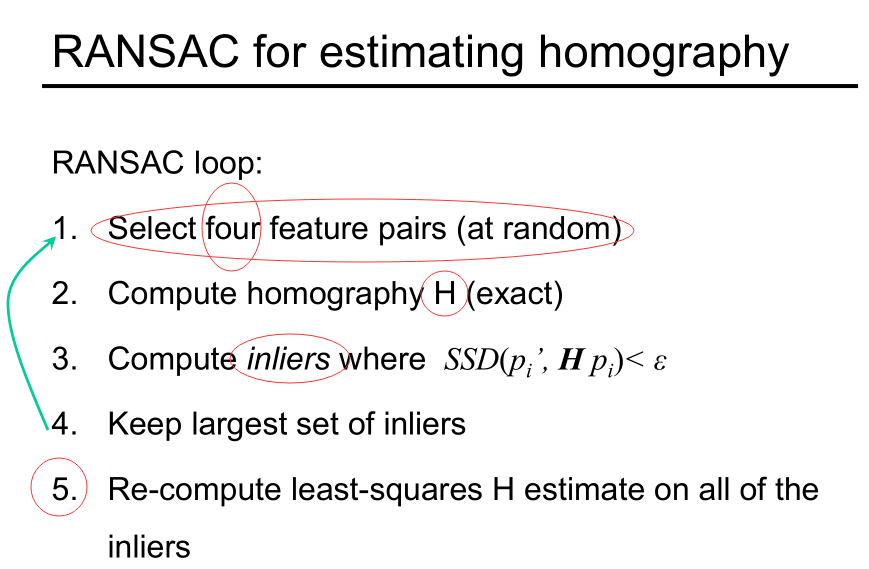
1. 对待拼接的两幅图像的sift特征点进行匹配：

根据lowe在论文里的算法描述，sift匹配采用最近邻和次近邻的sift匹配方法。取一幅图像中的一个SIFT关键点，并找出其与另一幅图像中欧式距离最近的前两个关键点，在这两个关键点中，如果最近的距离除以次近的距离得到的比率ratio少于某个阈值T，则接受这一对匹配点。阈值T越低，match的点越少但是越稳定，本次实验采用的阈值为0.6。代码如下：



1. 采用ransac算法计算带拼接图像需要做变形的矩阵：

实验代码采用的算法与课件上的一致：

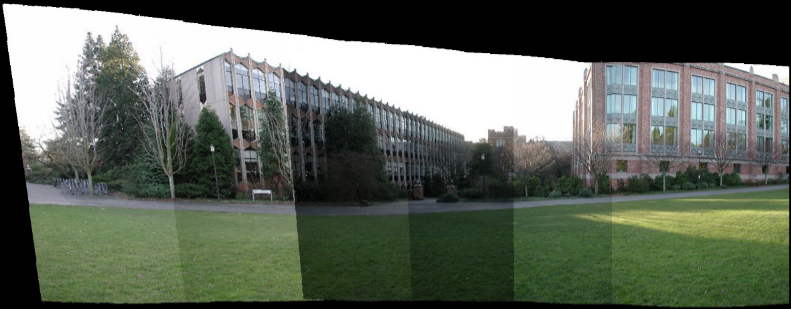


算法的思想是，在匹配的特征点中随机取四对点，求一个透视变换的矩阵。然后利用该矩阵，计算所有特征点对，通过该矩阵变换，依然为inliner的个数。通过不断地取样，循环，得出inliner最多的变换矩阵H，并用inliner的点所对应的match做最小二乘法求出更为精确的矩阵H，作为最终的变换矩阵。由于最小二乘法在C中比较复杂，且不做这一步效果并不会差多少，故C的代码里省略了这一步而matlab代码里实现了这一步。

此处代码太长不予展示。

1. 根据变换矩阵对两幅图像进行拼接：

图像拼接部分，一开始只是简单地将图片放上去然后覆盖，效果如下：



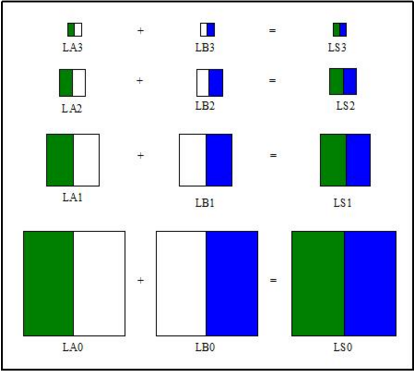
接缝处有很明显的突变。经过和同学的讨论后采用了一种叫Laplacian Pyramid Blending的方法。该方法算法如下：

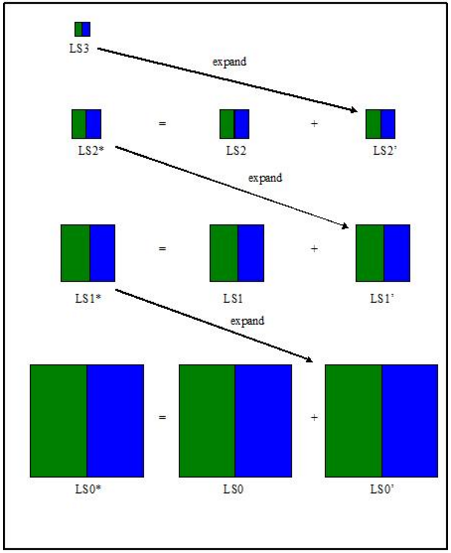
Pyramid Blening:

Laplacian pyramid is an algorithm using Gaussian to blend the image while keeping the significant feature in the mean time. It downsizes the image into different levels (sizes) with Gaussian. Later it expands the Gaussian in to the lower lever and subtracts from the image in that lever to acquire the Laplacian image.



After generating Laplacian pyramids for the overlap images A and B, we combine the two images in different Laplacian levels by combining partial images from each of them.





Afterward, we expand the LS from the top level () to the next level (N-1) and add it to the original Laplacian image in the corresponding layer () to generate the latest Laplacian image in the corresponding layer (). We repeat this step until reaching ground level () and the final result will be the blending image

算法来源于一个外国课程的介绍。其基本思想还是通过高斯差分金字塔来对图像的特征进行保留和重建。算法比较复杂，具体的意义不是很明白，然后照着算法实现了一个对应的算法。

此处代码太长不予展示。

1. 由于第一步的warping，图像到此步骤上下会留出一片无用的黑色区域，故在此对其进行判断并做了一个crop

至此，两幅图像拼接的算法结束，多幅图像拼接道理相同。

1. 实验结果及分析

对dataset1的4张图的拼接，结果如下：

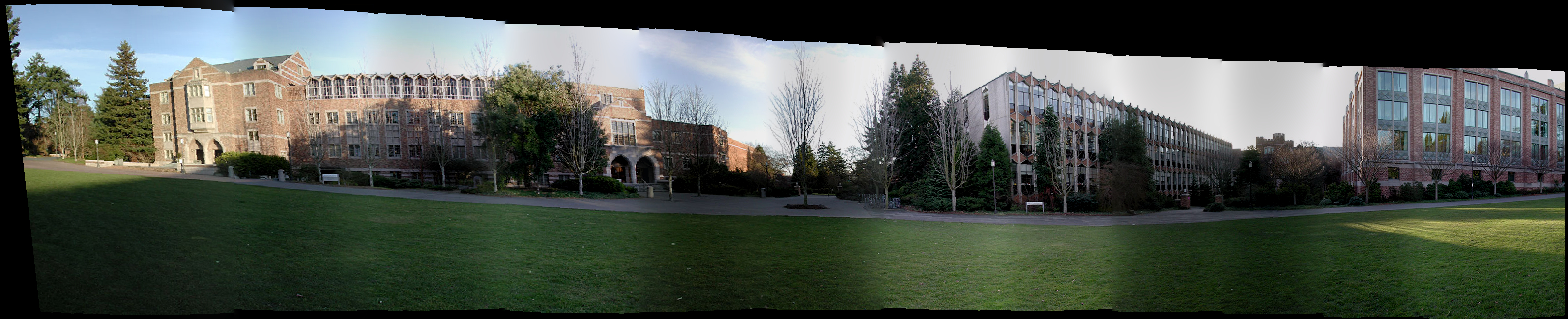


对dataset2的18张图的拼接，结果如下（包括中间结果）：

4张：



12张



全部18张：



1. 总结与反思

这次试验过程中跟老师说的一样，很多坑。本次的实验C++的代码基本上都是自己从头到尾实现的，并没有找一份现成项目进行修改和提交，也没有直接使用opencv这种成熟的库。Matlab的代码参考了网上一份代码的方法自己重新实现了一遍。在解决算法的过程中的各种坑让我对这整个实验的每个步骤有了更深刻的理解。并且，用了两种实现语言（C++和Matlab），两者在编程实现上各有特点（其中matlab在blending上没有做Laplacian Pyramid Blending），从中也学到了很多。

不足是，最终的结果在视觉上没有进一步的修正。