计算摄影学 Lab8 – 全景图拼接

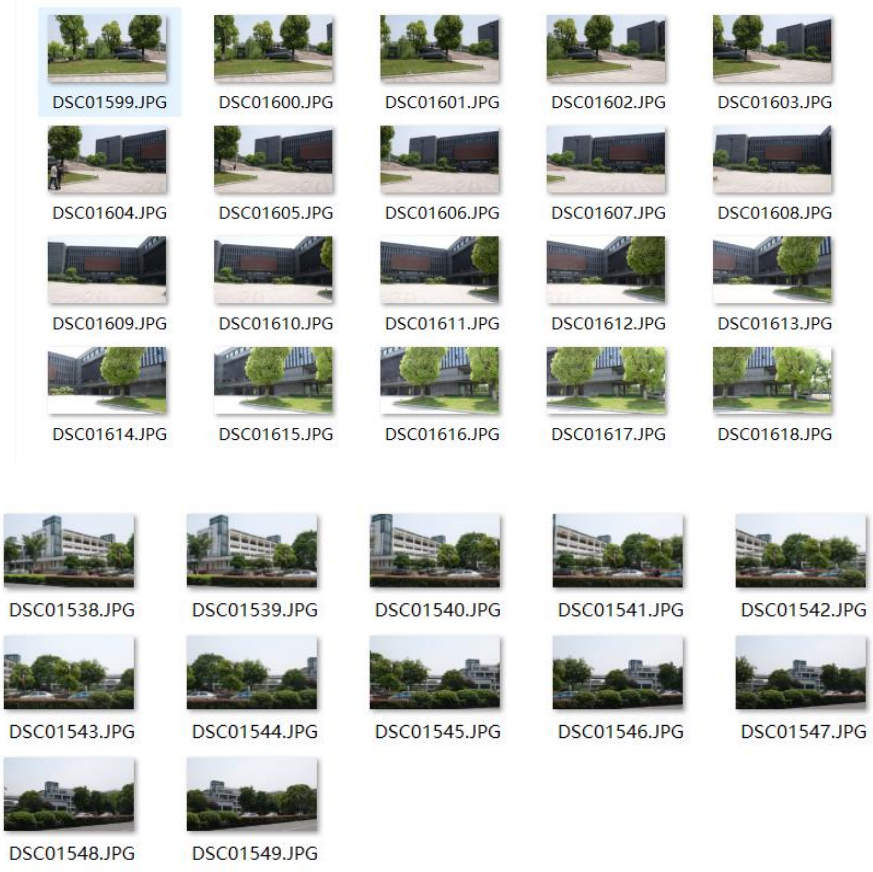
姓名：葛帅琦
学号：3150102193

一、实验背景

全景图拼接是利用同一场景的多张图像通过重叠部分寻找匹配关系，从而生成整个场景图像的技术。 全景图的拼接方法有很多，如按场景和运动的种类可以分为单视点全景拼接和多视点全景拼接。对于平面场景和只通过相机旋转拍摄的场景来说，可以使用求每两幅图像之间的一个 Homography 变换来映射到一张图像的方法，还可以使用恢复相机的旋转的方式得到最终的全景图。当相机固定只有水平方向旋转时，也可以使用柱面或球面坐标映射的方式求得全景图。

二、实验任务

本次试验获得两组数据（分别为 12,20 张连续拍摄的图片），目标是将图片利用 SIFT 特征匹配的方式生成新的全景图。



二、实验原理

SIFT 特征提取分析

声明：以下内容并非原创，很多地方我也看不懂，但是个人感觉很震撼。

SIFT（Scale-invariant feature transform）是一种检测局部特征的算法，该算法通过求一幅图中的特征点（interest points,or corner points）及其有关 scale 和 orientation 的描述子得到特征并进行图像特征点匹配，SIFT 特征不只具有尺度不变性，即使改变旋转角度，图像亮度或拍摄视角，仍然能够得到好的检测效果。整个算法分为以下几个部分：

1. 构建尺度空间

这是一个初始化操作，尺度空间理论目的是模拟图像数据的多尺度特征。

高斯卷积核是实现尺度变换的唯一线性核，于是一副二维图像的尺度空间定义为：

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

其中 G(x,y, σ) 是尺度可变高斯函数

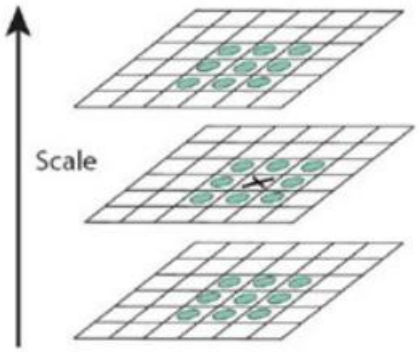
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

（x，y）是空间坐标，是尺度坐标。σ 大小决定图像的平滑程度，大尺度对应图像的概貌特征，小尺度对应图像的细节特征。大的 σ 值对应粗糙尺度(低分辨率)，反之，对应精细尺度(高分辨率)。为了有效的在尺度空间检测到稳定的关键点，提出了高斯差分尺度空间（DOG scale-space）。利用不同尺度的高斯差分核与图像卷积生成。

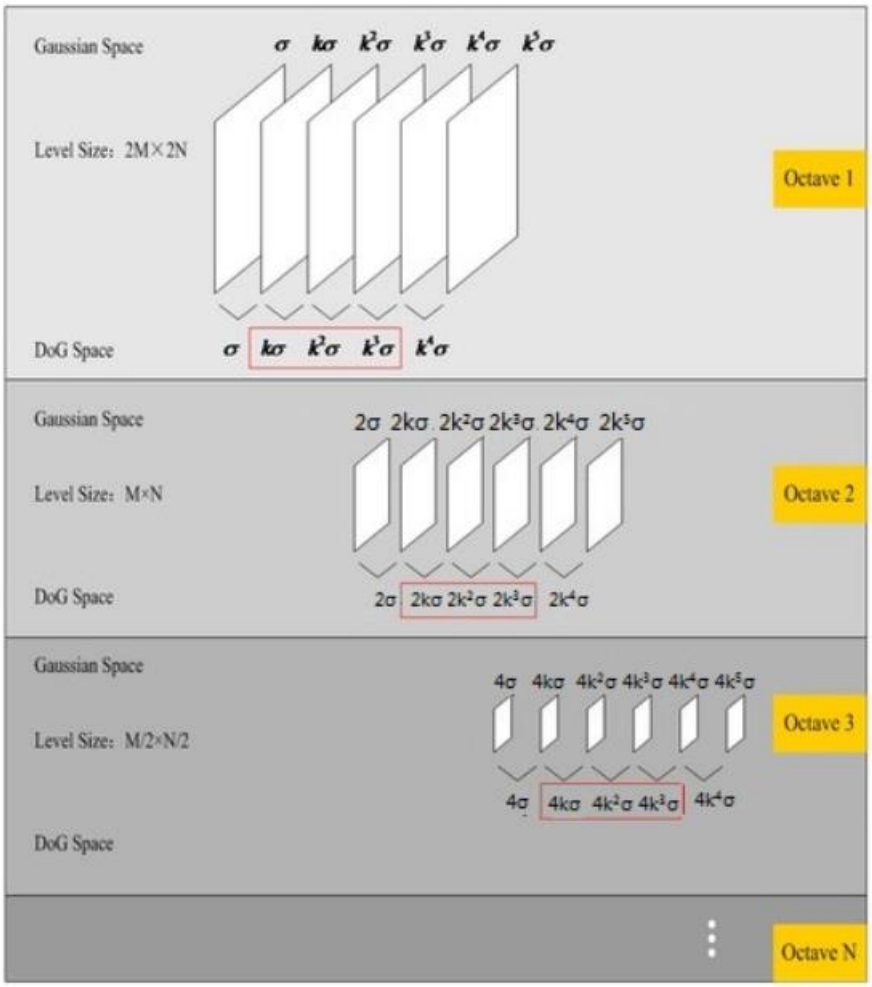
$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned}$$

2. LoG 近似 DoG 找到关键点<检测 DOG 尺度空间极值点>

为了寻找尺度空间的极值点，每一个采样点要和它所有的相邻点比较，看其是否比它的图像域和尺度域的相邻点大或者小。如图所示，中间的检测点和它同尺度的 8 个相邻点和上下相邻尺度对应的 9×2 个点共 26 个点比较，以确保在尺度空间和二维图像空间都检测到极值点。 一个点如果在 DOG 尺度空间本层以及上下两层的 26 个领域中是最大或最小值时，就认为该点是图像在该尺度下的一个特征点,如图所示。



同一组中的相邻尺度（由于 k 的取值关系，肯定是上下层）之间进行寻找



s=3的情况

3. 除去不好的特征点

这一步本质上要去掉 DoG 局部曲率非常不对称的像素。

通过拟和三维二次函数以精确确定关键点的位置和尺度（达到亚像素精度），同时去除低对比度的关键点和不稳定的边缘响应点(因为 DoG 算子会产生较强的边缘响应)，以增强匹配稳定性、提高抗噪声能力，在这里使用近似 Harris Corner 检测器。

空间尺度函数泰勒展开式如下：

$$D(\mathrm{x}) = D + \frac{\partial D}{\partial \mathrm{x}}^{T} \mathrm{x} + \frac{1}{2}\mathrm{x}^{\mathrm{T}} \frac{\partial^2 D}{\partial \mathrm{x}^2}\mathrm{x}$$

对上式求导,并令其为 0,得到精确的位置，得

$$\hat{\mathrm{x}} = -\frac{\partial^2 D}{\partial \mathrm{x}^2}^{-1} \frac{\partial D}{\partial \mathrm{x}}.$$

在已经检测到的特征点中,要去掉低对比度的特征点和不稳定的边缘响应点。去除低对比度的点：把公式(2)代入公式(1)，即在 DoG Space 的极值点处 D(x)取值，只取前两项可得：

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}.$$

若 $|D(\hat{x})| \geq 0.03$，该特征点就保留下来，否则丢弃。

边缘响应的去除

一个定义不好的高斯差分算子的极值在横跨边缘的地方有较大的主曲率，而在垂直边缘的方向有较小的主曲率。主曲率通过一个 2×2 的 Hessian 矩阵 H 求出:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

导数由采样点相邻差估计得到。

D 的主曲率和 H 的特征值成正比，令 α 为较大特征值， β 为较小的特征值，则

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

令 α = γ β，则

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

(r + 1)2/r 的值在两个特征值相等的时候最小，随着 r 的增大而增大，因此，为了检测主曲率是否在某域值 r 下，只需检测

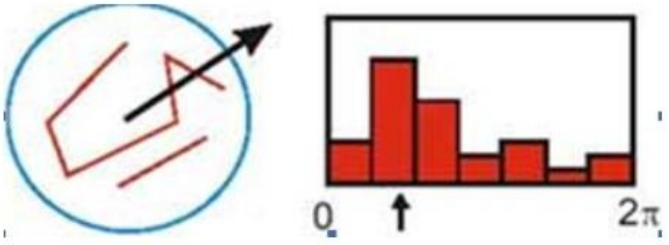
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

if (α + β)/ α β > (r+1)2/r, throw it out.　在 Lowe 的文章中，取 r＝10。

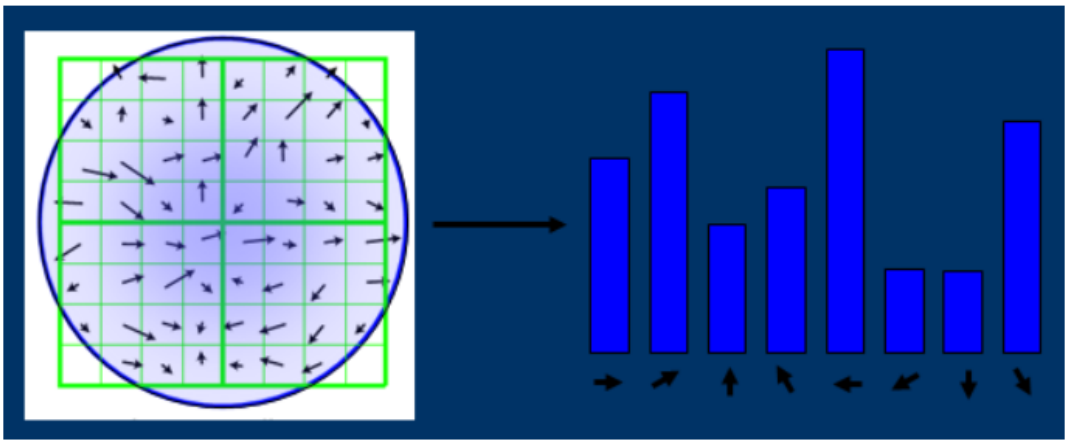
4. 给特征点赋值一个 128 维方向参数

上一步中确定了每幅图中的特征点，为每个特征点计算一个方向，依照这个方向做进一步的计算， 利用关键点邻域像素的梯度方向分布特性为每个关键点指定方向参数，使算子具备旋转不变性。

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = a\tan 2((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

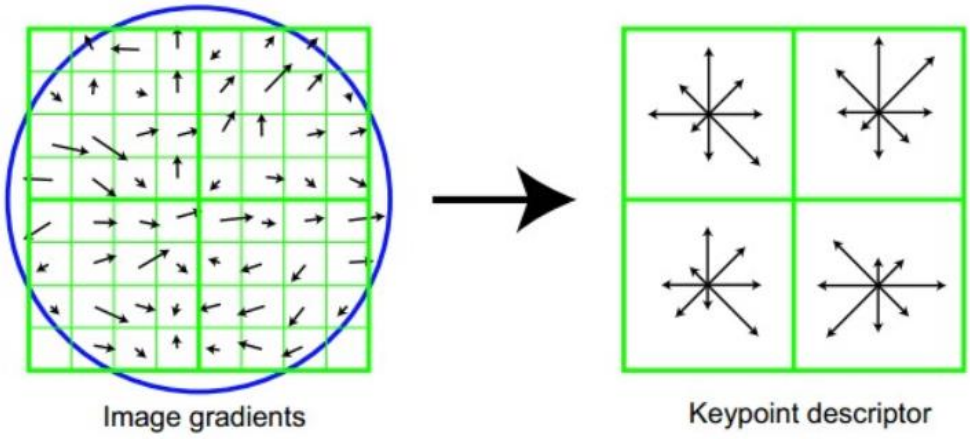


直方图中的峰值就是主方向，其他的达到最大值 80%的方向可作为辅助方向



5. 关键点描述子的生成

首先将坐标轴旋转为关键点的方向，以确保旋转不变性。以关键点为中心取 8×8 的窗口。



5. 根据 SIFT 进行 Match

生成了 A、B 两幅图的描述子，（分别是 k1*128 维和 k2*128 维），就将两图中各个 scale（所有 scale）的描述子进行匹配，匹配上 128 维即可表示两个特征点 match 上了。当两幅图像的 SIFT 特征向量生成后，下一步我们采用关键点特征向量的欧式距离来作为两幅图像中关键点的相似性判定度量。取图像 1 中的某个关键点，并找出其与图像 2 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少于某个比例阈值，则接受这一对匹配点。降低这个比例阈值，SIFT 匹配点数目会减少，但更加稳定。

柱面坐标的变换

对于每一幅图像来说，我们都可以把它们投影到一个柱面上，得到柱面上的图像 。柱面图像的坐标变换为

$$x' = r\tan^{-1}\left(\frac{x}{f}\right)$$

$$y' = \frac{ry}{\sqrt{x^2+f^2}}$$

其中(x′ ,y′)为柱面上的坐标，(x,y)为平面图像坐标，其坐标原点都已移至图像中心，r 为柱面半径，f 为焦距。然而为了得到柱面投影图像，我们往往需要将柱面图像上的点逆变换到平面图像上的对应像素点，进行插值，得到完整的柱面图像，逆变换的变换公式为

$$x = f\tan\left(\frac{x'}{r}\right)$$

$$y = \frac{y'}{r}\sqrt{x^2+f^2}$$

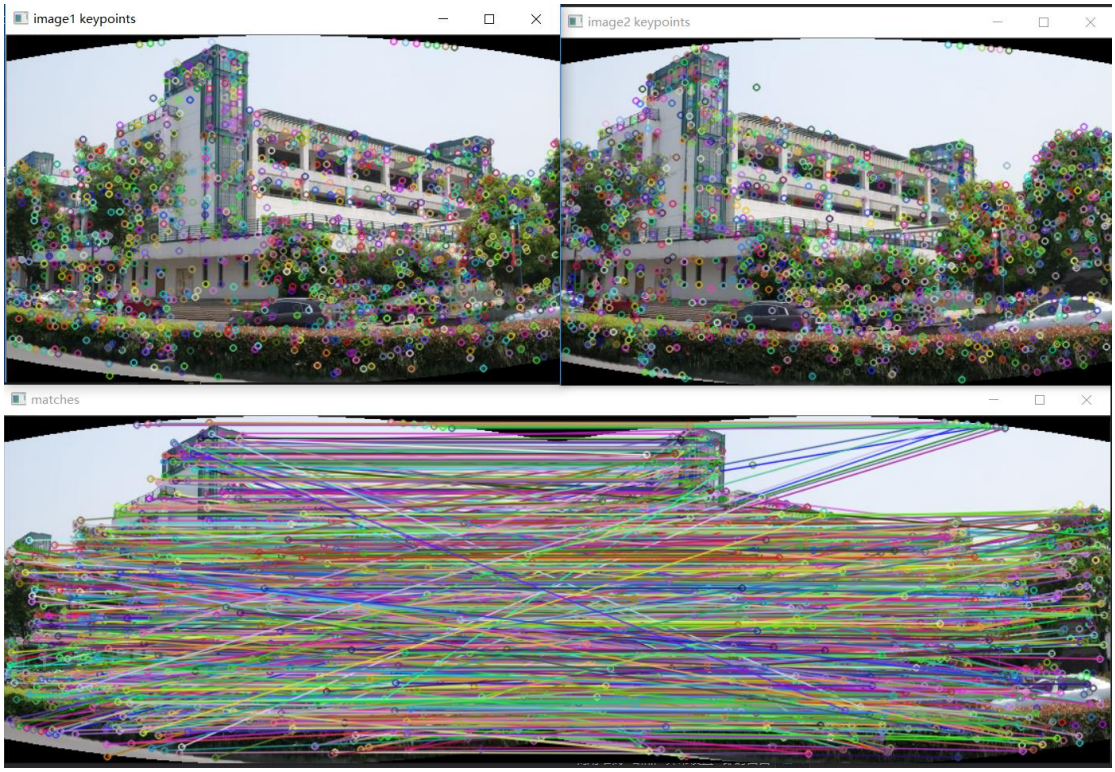
三、实验过程

1. 将图片根据上述数学公式做椭圆变换

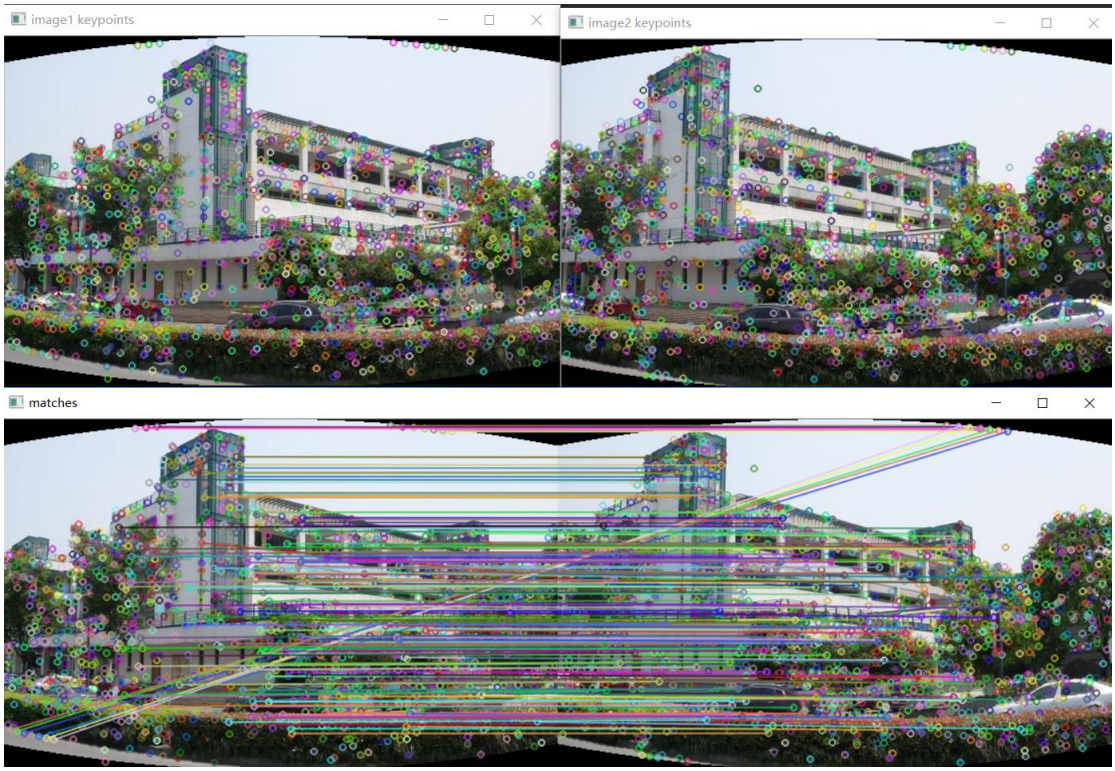
变换后

2. 利用 opencv 库中 SIFT 算法提取特征点

3. 抽取特征点描述做近似度匹配

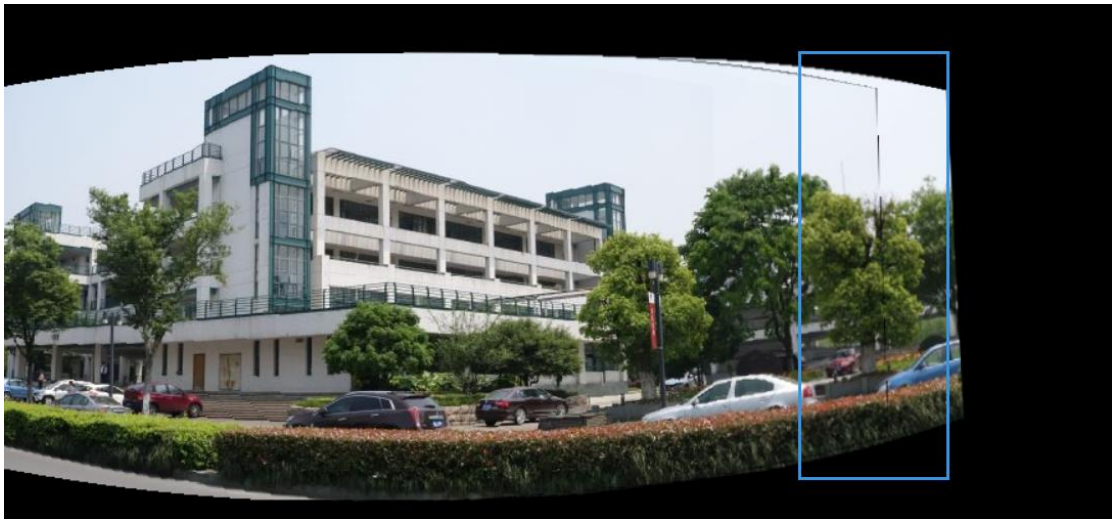


4. 不变点数量过多，看了一些教程后得知只需要取最佳匹配的部分点即可，因而我只取了最佳匹配的 200 个点



5. 找到 match 之后利用 findHomograph（）函数计算变换矩阵进行变换

6. 然后将两张图合并



7. 轮廓线的消除

当我合并多张图的时候，我发现黑色轮廓线很明显，原因可能是 SIFT 库处理完图形之后，在边缘的像素可能本身偏暗。所以之后的代码采用比较 RGB 范数的形式来判定是否需要替换像素。

```
    // We can do this since im_1 and wim_2 have the same size.
    for (int i = 0; i < img.cols; i++)
        for (int j = 0; j < img.rows; j++) {
            Vec3b color_im1 = img.at<Vec3b>(Point(i, j));
            Vec3b color_im2 = wim_2.at<Vec3b>(Point(i, j));
            // 新
            if (norm(color_im1) < norm(color_im2))
                img.at<Vec3b>(Point(i, j)) = color_im2;
            // 旧
            //if (norm(color_im1) == 0)
            //   img.at<Vec3b>(Point(i, j)) = color_im2;

        }
```

得到的结果很理想。

8. 最终拼接全景图,

拼接时，为了让图像尽可能的对称，选择从中间开始拼器，然后依次左右添加图片，而不是从最左直接遍历到最右边。例如有从左到右依次有5张图片1,2,3,4,5, 拼接顺序为：

img <- 3

img <- 2 + img

img <- img + 4

img <- 1 + img

img <- img + 5

第二章图虽然照片形变的很厉害，但是景物看起来还是比较自然的。里面的人由于人在移动，一部分相片有人，一部分没人，导致人的拼接效果不好。

核心代码

```
class Oanorama2193: public CylindricalPanorama {
    double r = 500;
public:
    vector<Mat> image;
    void readImage(vector<string> name, double f);
    bool makePanorama(vector<Mat>& img_vec, Mat& img_out, double f);
    void imagemerge(Mat& img, Mat& img2);
    Mat ScalaLarge(Mat img, int rows, int cols);
    void displayAllImage();
    Mat project2cylinder(Mat image, double r, double f);
    void checkborder(Mat &img);
};
void readImage(vector<string> name, double f); // 读取图像
bool makePanorama(vector<Mat>& img_vec, Mat& img_out, double f); // 全景图拼接
void imagemerge(Mat& img, Mat& img2); // 合并两张图
Mat ScalaLarge(Mat img, int rows, int cols); //所有图片放大到相同大小
void displayAllImage(); // 显示读到的图片
Mat project2cylinder(Mat image, double r, double f); // 投影柱面
```

总结

这次实验算是助教给的实验指导最不详细的一次实验了，也是难度最大的一次小作业。第一次用opencv里面从来没用过的SIFT库，做出了在论文里面才能看见的一些效果图，感觉很不错。学到了很多东西。但是回过头来反思吧，自己并没有足够的数学能力来理解关于SIFT的证明以及推导，还是数学比较重要。David Lowe老爷子太强了。