

全景拼接实验报告

成员

龚逸青：201712809008

方家琦：201712512018

实验环境

- python 3.6
 - opencv-python 3.4.2.16
 - numpy 1.18.1

实现流程

我们采用以下流程实现单视点的全景拼接：

- 使用sift算法找出每张图片的特征点
- 使用 `opencv` 的 `knnmatch` 匹配图片之间的特征点
- 使用 `ransac` 方法确定合理的特征匹配，同时求出 `homography` 矩阵。
- 根据 `homography` 使用双线性插值将一张图转到另一张图的坐标系中。
- 使用合适的 `blending` 方法将两张图拼接在一起。

ransac求解homography

RANSAC是“Random Sample Consensus (随机抽样一致) ”的缩写。它可以从一组包含“局外点”的观测数据集中，通过迭代方式估计数学模型的参数。它是一种不确定的算法——它有一定的概率得出一个合理的结果；为了提高概率必须提高迭代次数。

1. 数据由“局内点”组成，例如：数据的分布可以用一些模型参数来解释；
2. “局外点”是不能适应该模型的数据；
3. 除此之外的数据属于噪声。

对于我们在全景拼接中的求解 `homography` 的任务，我们可以应用 `ransac` 的方法来求解。

先从匹配点对中随机选择 n 个点作为确实匹配的点对，根据这 n 个点使用最小二乘法可以求解出一个转换矩阵，使用这个转换矩阵可以求出在一定误差范围内符合这个转换矩阵的原匹配点对数，原匹配点对数越多，说明选出的 n 个点全是是“局内点”的概率越高。这样不断迭代，就有很大的概率取到最优解。

我们可以从理论上推导迭代次数和渠道最优解概率的关系。

当我们从估计模型参数时，用 p 表示一些迭代过程中从数据集内随机选取出的点均为局内点的概率；此时，结果模型很可能有用，因此 p 也表征了算法产生有用结果的概率。用 w 表示每次从数据集中选取一个局内点的概率，如下式所示：

$$w = \text{局内点的数目} / \text{数据集的数目}$$

通常情况下，我们事先并不知道 w 的值，但是可以给出一些鲁棒的值。假设估计模型需要选定 n 个点，

w^n 是所有 n 个点均为局内点的概率； $1 - w^n$ 是 n 个点中至少有一个点为局外点的概率，此时表明我们从数据集中估计出了一个不好的模型。 $(1 - w^n)^k$ 表示算法永远都不会选择到 n 个点均为局内点的概率，它和 $1 - p$ 相同。因此，

$$1 - p = (1 - w^n)^k$$

我们对上式的两边取对数，得出

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

值得注意的是，这个结果假设 n 个点都是独立选择的；也就是说，某个点被选定之后，它可能会被后续的迭代过程重复选定到。这种方法通常都不合理，由此推导出的 k 值被看作是选取不重复点的上限。

当我们从估计模型参数时，用 p 表示一些迭代过程中从数据集中随机选取出的点均为局内点的概率；此时，结果模型很可能有用，因此 p 也表征了算法产生有用结果的概率。用 w 表示每次从数据集中选取一个局内点的概率，如下式所示：

$$w = \text{局内点的数目} / \text{数据集的数目}$$

通常情况下，我们事先并不知道 w 的值，但是可以给出一些鲁棒的值。假设估计模型需要选定 n 个点， w^n 是所有 n 个点均为局内点的概率； $1 - w^n$ 是 n 个点中至少有一个点为局外点的概率，此时表明我们从数据集中估计出了一个不好的模型。 $(1 - w^n)^k$ 表示算法永远都不会选择到 n 个点均为局内点的概率，它和 $1 - p$ 相同。因此，

$$1 - p = (1 - w^n)^k$$

我们对上式的两边取对数，得出

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

值得注意的是，这个结果假设 n 个点都是独立选择的；也就是说，某个点被选定之后，它可能会被后续的迭代过程重复选定到。这种方法通常都不合理，由此推导出的 k 值被看作是选取不重复点的上限。

坐标系转换

在使用 `ransac` 求出转换矩阵后我们需要将一张图片的坐标系进行转换。在转换的过程中，主要涉及以下两方面的问题：

- 转换方向
 - 正向转换
 - 逆向转换
- 重采样方式
 - 取最邻近点
 - 线性插值
 - 高斯核过滤

转换方向

在转换过程中，如果我们采用正向计算的方式，即根据原图中的坐标计算其在新图中的坐标。那么我们会在新图中，很有可能会遇到新图中的某些点对应于原图中的多个点，而有些点则没有在原图中的对应点，导致值得缺失。

因此，我们采用逆向转换的方式来得到新图。

我们先确定想要得到新图大小。对于新图中得每个像素点，根据 `homography` 的逆矩阵计算其在原图中的对应点。


由于这样计算出来的坐标不一定为整数(大概率不是)，所以我们还需要考虑如何取像素点的问题。

重采样方式

最简单的重采样方式就是直接取最邻近点的像素值。这个方法的优点是效率高，但是相应的，效果不是很好，转换过后的图中可能会有比较明显的锯齿。

另一个方法是采用高斯核进行过滤，这样可以得到一个比较平滑的结果图，但是使用高斯核会导致计算量大大增加。

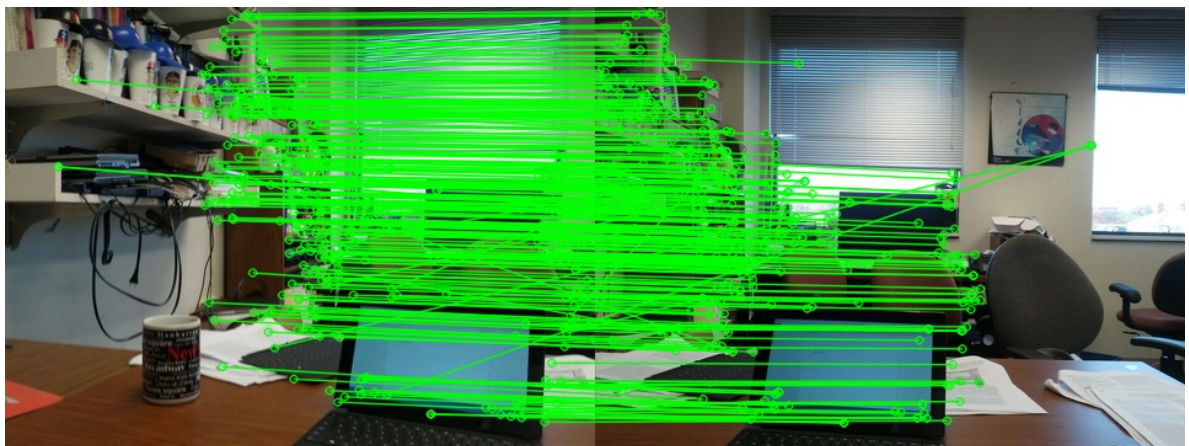
所以我们采用一个折中的方式，使用双线性插值。

 双线性差值

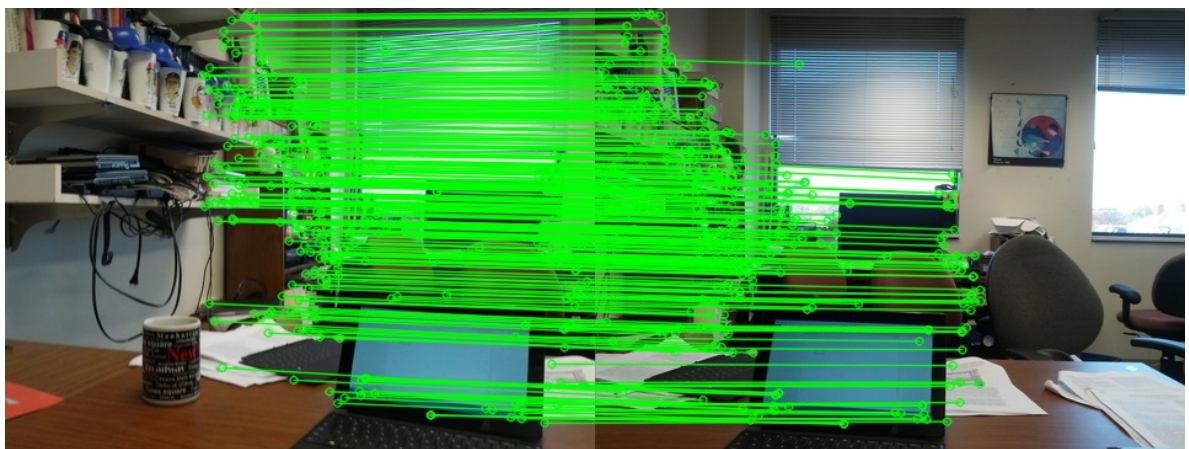
实验结果分析

ransac求解homography

利用ransac方法，首先对SIFT特征抓取得到的匹配点展开单映性矩阵的计算，其对于噪声的筛选效果如下所示：



图一 利用SIFT计算得到的correspondences



图二 利用ransac计算出的homography对应内点集

可以看到所有的内点都近乎平行，这体现了阈值设定的效果。而可以看到，内点的数量相当可观，这也印证了计算出的单映性矩阵性值良好。

坐标系转换

接着，我们使用warp()方法对需要被映射的图片进行仿射变换，其效果如下图所示：

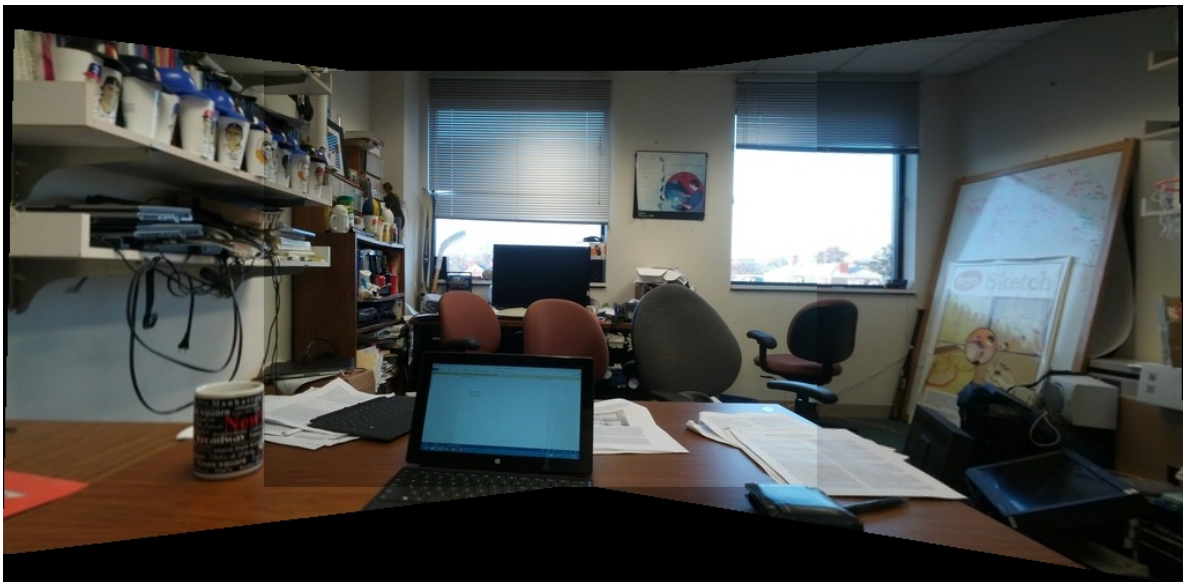


图三 经过warp方法得到的映射后的图片

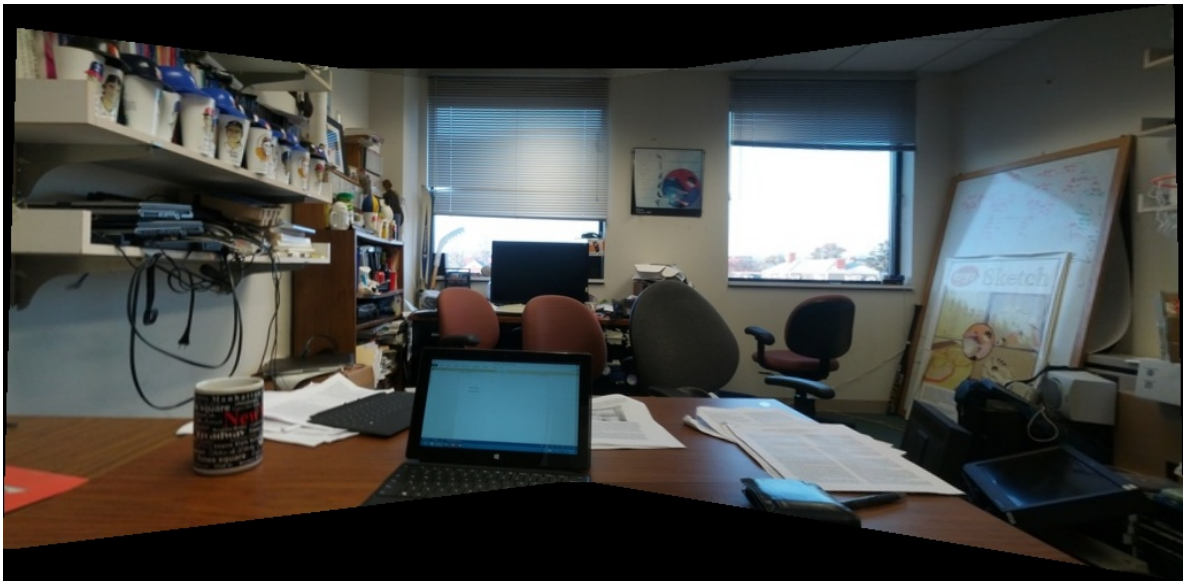
由于这一方法的优秀约束，变换后的图片边缘并没有与纯黑的像素点进行双线性差值，这样映射得到的图形边缘非常干净，为下一步的blending打下了非常好的基础。

图像拼接

该步骤对于图像做了一个简单的线性权重处理，其效果比直接拼接好很多，但仍有不足。



图四 无处理的直接拼接



图五 带权重的拼接

可以看到，其效果相对而言平滑很多，在横向上看结果的性能还不错。