数字图像处理综合论文

(Digital Image Processing Thesis)

论文题目 基于特征点匹配的图像拼接技术研究

作者姓名 周楠

专　　业 自动化

学　　号 2014011472

指导老师 陆文凯

2016 年 10 月 29 日

目录

摘要 4

Abstract 5

第一章　绪论 6

1.1 课题背景 6

1.2 发展动态 6

1.2.1 国外发展现状 6

1.2.2 国内发展现状 7

1.3 本文主要内容 7

第二章　手动兴趣点匹配的拼接算法 8

2.1 理论分析 8

2.2 透视变换矩阵的求解算法 8

2.3 透视变换的结果 11

2.4 图像融合算法 12

2.5 图像融合算法的效果 13

2.6 渐变的图像融合算法 15

2.7 渐变的图像融合算法的效果 16

2.8 频域低通滤波算法 17

2.9 频域低通滤波的效果 18

2.10 本章小结 18

第三章　基于 Harris 角点匹配的拼接算法 19

3.1 理论分析 19

3.2 Harris 角点探测算法 21

3.3 Harris 角点探测算法的效果 22

3.4 Harris 角点匹配算法 24

3.5 Harris 角点匹配算法的效果 25

3.6 图像融合算法及效果 27

3.7 手动算法和 Harris 角点算法的比较 27

3.8 本章小结 29

第四章　基于 Surf 特征点匹配的拼接算法 30
4.1 理论分析 30
4.2 基于 Surf 特征点匹配的拼接算法及效果 30
第五章　三种算法的优劣对比 34
5.1 效果对比 34
5.2 算法复杂度对比 35
5.3 算法速度对比 36
第六章　总结与展望 37

摘要

本文基于特征点的匹配的思想，用手动兴趣点匹配、Harris 角点匹配、Surf 特征点匹配三种方法，基于 Matlab 编程，实现了三幅由智能手机拍摄的数字图像的拼接。其中，手动兴趣点匹配、Harris 角点匹配全部函数的编程工作由自己完成，Surf 特征点匹配的方法借用了 Matlab 的库函数工具包。本文进行了详细的文献调研和网络资源整合，整理、分析了三种方法的原理，并且对其性能以及效果进行了分析和比较。此外，为了提高拼接效果，本文对接缝处进行了渐进取值，并利用频域低通滤波器，对拼接后的图像进行了平滑处理。

关键词： 图像拼接，图像融合，透视变换，特征点，Harris 角点，Surf 算法

Abstract

In this paper, based on the idea of feature points matching, we use three methods to finish the image stitching assignment, which contains feature points matching by human, Harris method and Surf algorithm. We use the software Matlab to program, which successfully finish the stitching project. Among the three methods, the first way and the second way are all programmed by ourselves. Surf feature points matching method contains some tool functions which are already finished by Matlab. This paper makes a detailed literature survey and resource integration. Moreover, the performance and effect of these methods are analyzed and compared. At last, in order to improve the effect of stitching, we use the frequency domain low-pass filter which could smooth the final picture after stitching.

Keywords: image stitching, image fusion, perspective transformation, feature points, Harris method, Surf algorithm

第一章　绪论

1.1 课题背景

随着便携式数码照相设备的普及，传统的胶片式照相技术被逐渐淘汰，随之产生的，便是越来越多的数字图像出现在人们的生活中。在一些情形下，人们需要使用普通的照相机甚至是智能手机来获取宽视野的场景图像，传统的方法是调节相机焦距，以牺牲分辨率的方法增大视野；此外，在一些科研领域，特别是航天和遥感领域[1]，往往需要获取超大尺寸的物体的全景图像，广角照相机可以部分地解决问题，但其价格昂贵，且在图像边缘容易出现“鱼眼”类型的畸变。

随着计算机技术的发展，数字图像拼接技术很好地解决了这类问题。其能在软件上，把若干幅小视角的图像中重叠的部分融合在一起，从而拼接成一幅全景图像。数字图像拼接技术具有鲁棒性高、成本低、可重复使用的特点，从而被广泛使用。

随着越来越多优秀的拼接算法的提出，数字图像拼接技术愈发成熟，其使用范围也被拓展。虚拟现实[2]、医学成像、模式识别，甚至是公安取证和军事领域，都在使用这一技术。

由此可见，图像拼接技术具有很好的应用前景和重要的科研意义。

1.2 发展动态

1.2.1 国外发展现状

最早的图像拼接技术是基于相位相关度实现的。1975 年，Kuglin 和 Hines 提出图像相位相关法[3]， 使用傅立叶变换将两幅待拼接的数字图像变换到频域，再使用互功率谱直接计算出两幅图像之间的平移量。随后 Reddy 和 Chaterji 改进了此算法，使得效率大大提高。随着快速傅里叶变换算法的提出，图像拼接技术得到很大的发展。

随后，出现了基于几何特征的图像配准方法。在 1988 年，Harris 研究出了 Harris 角点检测器，本文将自己实现 Harris 角点的检测和两幅图间的角点匹配。1994 年，Deriche 通过二维高

---

[1] 谭康. 图像拼接技术研究与实现[D]南京,2006

[2] 张茂军，虚拟现实系统[M]北京：科学出版社，2010

[3] C.Kuglin, D.Hines. The Phase Correlation Image Alignment Method[C]IEEE Conf. 1975

斯模糊过滤，得到了一些低级特征模型[4]。Leila M. G. Fonseca 在 1997 年提出一种基于小波变换的多尺度分析算法，他们首先对数字图像做小波变换，然后计算小波变换的模值，模的局部极大值点就是数字图像中的边缘特征点。后来，越来越多的特征点算法被提出，比较著名的还有 Sift 算法，以及后来对 Sift 算法优化后的 Surf 算法。本文将借助 Matlab，实现 Surf 特征点的寻找、匹配、误匹配点剔除的研究。

1.2.2 国内发展现状

国内图像拼接技术起步相对较晚，但发展迅速。2001 年，清华大学研究出了一种使用三角架固定相机进行拍摄的算法，来解决图像拼接中计算量与拼接精度的互相制约的情况。中国科学院自动化研究所的韦燕凤等人研究出一种利用网格结构图像的特点，进行目标配准定位的算法。中国科学院力学研究所的赵唯也实现了基于小波变换的图像拼接技术。

此外，许多高校先后开展了数字图像处理的课程，在网络上也可以检索到许多相关的技术性博客。可见，图像拼接技术的研究十分普遍。

1.3 本文主要内容

本文基于特征点的匹配的思想，用手动兴趣点匹配、Harris 角点匹配、Surf 特征点匹配三种方法，基于 Matlab 编程，实现了三幅由智能手机拍摄的数字图像的拼接。本文整理、分析了三种方法的原理，并且对其性能以及效果进行了分析和比较。此外，为了提高拼接效果，本文利用频域低通滤波器，对拼接后的图像进行了平滑处理。

下面是具体的章节安排：

第一章　简要介绍数字图像拼接技术的背景和国内外发展动态，以及本文的主要内容；

第二章　研究手动兴趣点匹配的拼接原理，并自己实现了算法与编程；

第三章　研究基于 Harris 角点匹配的拼接原理，并自己实现了算法与编程；；

第四章　研究 Surf 特征点匹配的拼接原理，借助 Matlab 工具包实现了算法和编程；

第五章　对比并分析三种方法的优劣；

第六章　本文研究结论与展望。

本文使用的编程软件为 Matlab R2015a。

[4] Deriche R. Recovering and characterizing image features using an efficient model based approach.[C] IEEE Computer Vision Conference. 1994

第二章　手动兴趣点匹配的拼接算法

2.1 理论分析

简单的来说，图像拼接算法分为两个部分，第一个部分便是图像配准，目的是对图像进行一系列的变换，使其相同的“特征部分”能够较好地重叠；第二个部分便是图像融合，把三幅已经处理好的图像放在一个坐标系下，算出每幅图的变换后的位置，进行融合即可。

2.2 透视变换矩阵的求解算法

手动兴趣点匹配的拼接算法的核心便是根据手动选取的点，算出两幅图之间的变换矩阵。不妨设三幅待拼接的图像为 A、B、C，而已知 B 是图像的中心，显然 A 与 B 的配准相似于 B 与 C 的配准，为了简单，下面只讨论前者。

想法是，若能在 A，B 中找到 4 对相同的点(每对点在最后的全景图中重合)，根据其坐标，便能算出变换矩阵。不妨设 A 中找到了四个点，其组成的坐标矩阵为 Moving_Points

Moving_Points=

x1	y1	1
x2	y2	1
x3	y3	1
x4	y4	1

在 B 中有四个点与之对应，其组成的坐标矩阵为 Fixed_Points

Fixed_Points=

x1’	y1’	1
x2’	y2’	1
x3’	y3’	1
x4’	y4’	1

目标是找到一个3×3的矩阵 T，使得

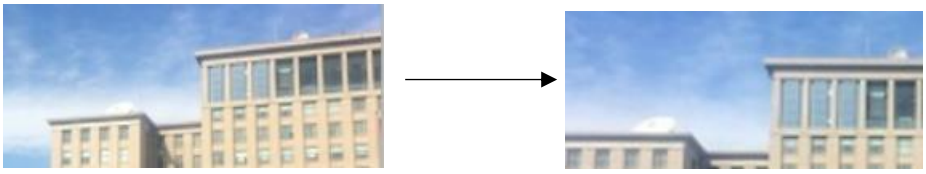
$$\text{Fixed\_Point} = \text{Moving\_Points} \times \text{T}$$

T 中有 9 个未知数，是一个标准的矩阵变换的定义方式，

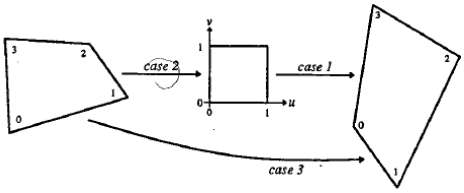
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

上诉变换矩阵可以拆成 4 部分，$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$表示线性变换，$\begin{bmatrix} a_{31} & a_{32} \end{bmatrix}$用于平移，$\begin{bmatrix} a_{13} & a_{23} \end{bmatrix}^T$产生透视变换，$a_{33}$通常为 1。

若列出线性方程组，总共可以建立 12 个方程组，必然是有解的。这样，就建立了 A 中的点到 B 中点的一一映射。当把图像 A 照此矩阵映射之后，A 中主楼倾斜的房顶就可以变为 B 中主楼水平的房顶。如下所示



这个变换就是图相变换中常用的透视变换， 不同于仿射变换，透视变换后的矩阵可以变为任意四边形。原图中平行线在新图中不一定保持，透视变换的效果如下：



这里我们采用奇异值分解的方法，算出这个矩阵 T。算法如下：

算法 2-1 透视变换矩阵的求解算法

输入：参考点数组 Fixed_Points[x,y]

x1’	y1’	1
x2’	y2’	1
x3’	y3’	1
x4’	y4’	1

，移动点 Moving_Points[x,y]

x1	y1	1
x2	y2	1
x3	y3	1
x4	y4	1

输出：变换矩阵 T

1. 计算新矩阵 A，新矩阵的元素如下

x1	y1	1	0	0	0	-x1*x1’	-y1*x1’	-x1’
0	0	0	x1	y1	1	-x1*y1’	-y1*y1’	-y1’
x2	y2	1	0	0	0	-x2*x2’	-y2*x2’	-x2’
0	0	0	x2	y2	1	-x2*y2’	-x2*y2’	-y2’
x3	y3	1	0	0	0	-x3*x3’	-y3*x3’	-x3’
0	0	0	x3	y3	1	-x3*y3’	-y3*y3’	-y3’
x4	y4	1	0	0	0	-x4*x4’	-y4*x4’	-x4’
0	0	0	x4	y4	1	-x4*y4’	-y4*x4’	-y4’

2. 对 A 奇异值分解；$A = U \times S \times V^T$；其中 V 是 A 的右奇异向量；

3. 取 V 的第一列，即第一个右奇异向量；

4.它的 9 个值分别幅值给 T，输出 T；

可以证明[5]，下式成立

$$\begin{bmatrix} kx1' \\ ky1' \\ k \end{bmatrix} = T \begin{bmatrix} x1 \\ y1 \\ 1 \end{bmatrix}$$

即 T 为移动点数组到参考点数组的透视变换的矩阵。

验证如下：

输入点 Fixed_Points[x,y]和 Moving_Points[x,y]，生成的 A 如下所示，

18.25	175.75	1	0	0	0	-2833.31	-27285.2	-155.25
0	0	0	18.25	175.75	1	-3189.19	-30712.3	-174.75
62.75	176.25	1	0	0	0	-12346.1	-34677.2	-196.75
0	0	0	62.75	176.25	1	-10808.7	-30359.1	-172.25
88.75	330.25	1	0	0	0	-20700.9	-77030.8	-233.25
0	0	0	88.75	330.25	1	-28644.1	-106588	-322.75
25.25	330.25	1	0	0	0	-4298.81	-56225.1	-170.25
0	0	0	25.25	330.25	1	-8010.56	-104772	-317.25

奇异值分解后，取出 V 的一列生成变换矩阵 T，如下，

0.7082	-0.2304	-0.00097
0.036221	0.85242	-0.0001
130.3645	22.84409	1

代入点的坐标，有$\text{Moving_Points} \times T =$

149.655	168.4522	0.963961
181.188	158.6258	0.920905
205.1792	283.9083	0.879654
160.2085	298.5384	0.941019

分别处以第三列进行归一化，结果为

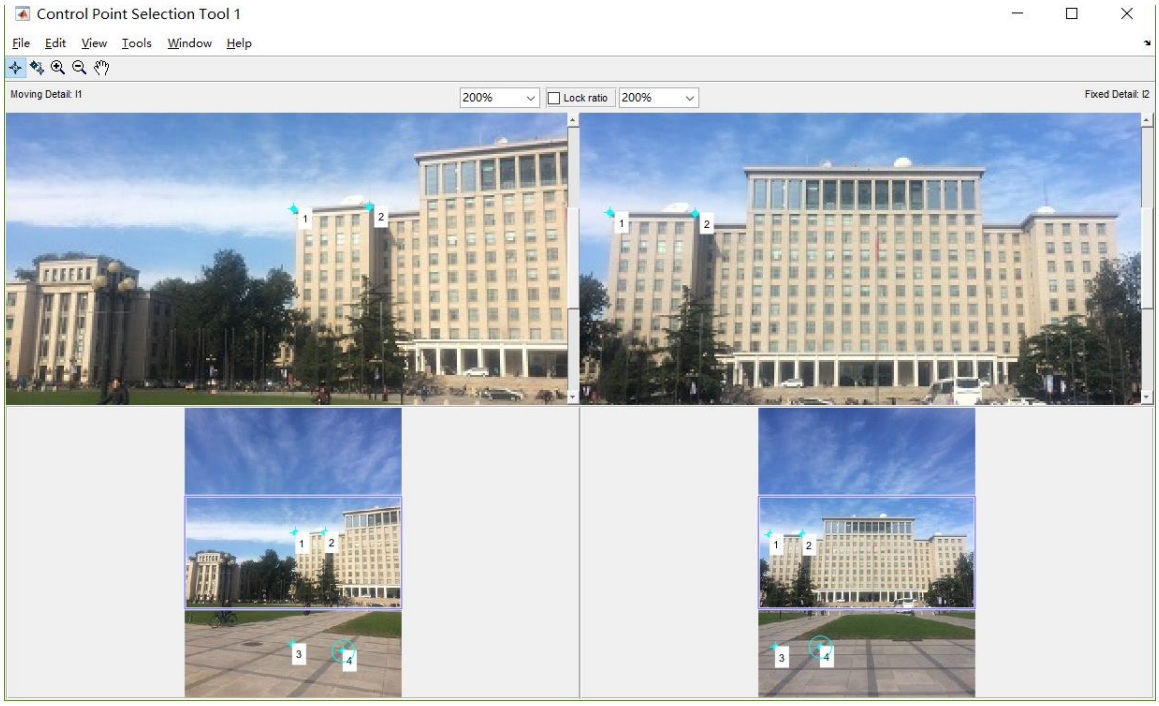
[5] http://blog.csdn.net/xiaowei_cqu/article/details/26471527

155.25	174.75	1
196.75	172.25	1
233.25	322.75	1
170.25	317.25	1

与 A 的最后一列相比，发现吻合度很高，说明变化矩阵效果很好。

2.3 透视变换的结果

用 cpselect 函数进行选点，选取了四对匹配点，如下图所示



透视变换前后的结果如下：

变换前，地板和主楼屋顶明显不平齐


变换后，地板和主楼屋顶变得平齐


2.4 图像融合算法

求得透视变换之后，A 到 B 的矩阵 T1，C 到 B 的矩阵 T3 均已经求出，那么可以根据这两个矩阵，求得透视变换之后的每幅图的范围。再生成一个全景图坐标系，在这个坐标系中，完成 A、B、C 三幅图的透视变换 A’、B’、C’。再让 B’图不动，直接把 A’、B’两幅图把它们与 B’不相同的地方粘上去。也就是说，全景图相当于

$$\text{panorama} = (A' - B') + B' + (B' - C')$$

具体算法如下

算法 2-2 图像融合算法
输入：透视变换矩阵 T1，T2(单位阵)，T3，图片 A，B，C 输出：全景图 D 1. 分别计算矩阵 A，B，C 在透视变换下的 x (y 方向同理，这里不再赘述) 方向坐标的最

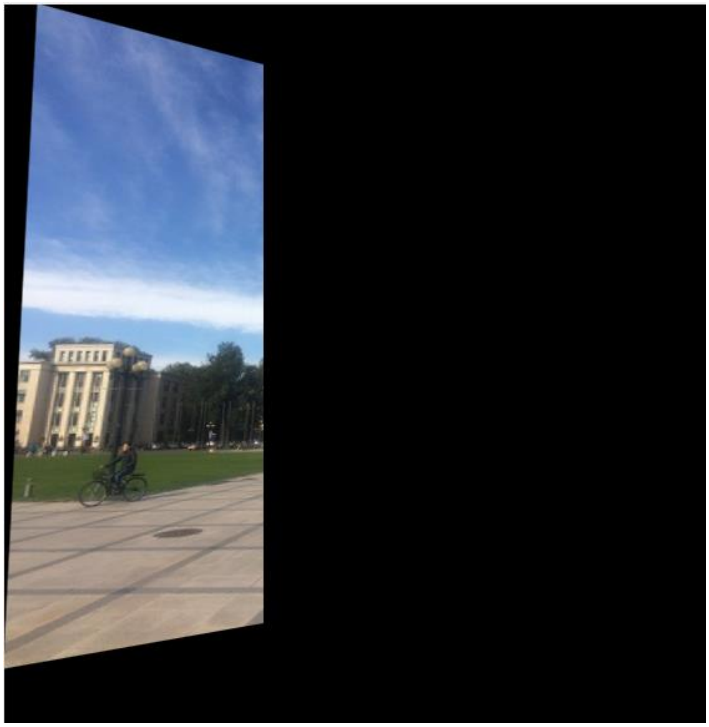
大值和最小值，分别记为[x_A_low, x_A_high],[x_B_low, x_B_high],[x_C_low, x_C_high]；

2. 选取 x_lowest=min{ x_A_low , x_B_low, x_C_low };
   选取 x_highest=max{ x_A_high , x_B_high, x_C_high };
3. 生成空的全景图，全景图的宽度为 x_highest-x_lowest；(高度为 y_highest-y_lowest)
4. 在全景图上，宽度从 1 到 x_B_low - x_A_low 处，放置 A 的投影图；
   宽度从 x_B_low - x_A_low 到 x_B_high- x_A_low 处，放置 B 的投影图；
   宽度从 x_B_high- x_A_low 到 x_C_high- x_A_low 处，放置 C 的投影图；
5. 输出全景图；

2.5 图像融合算法的效果

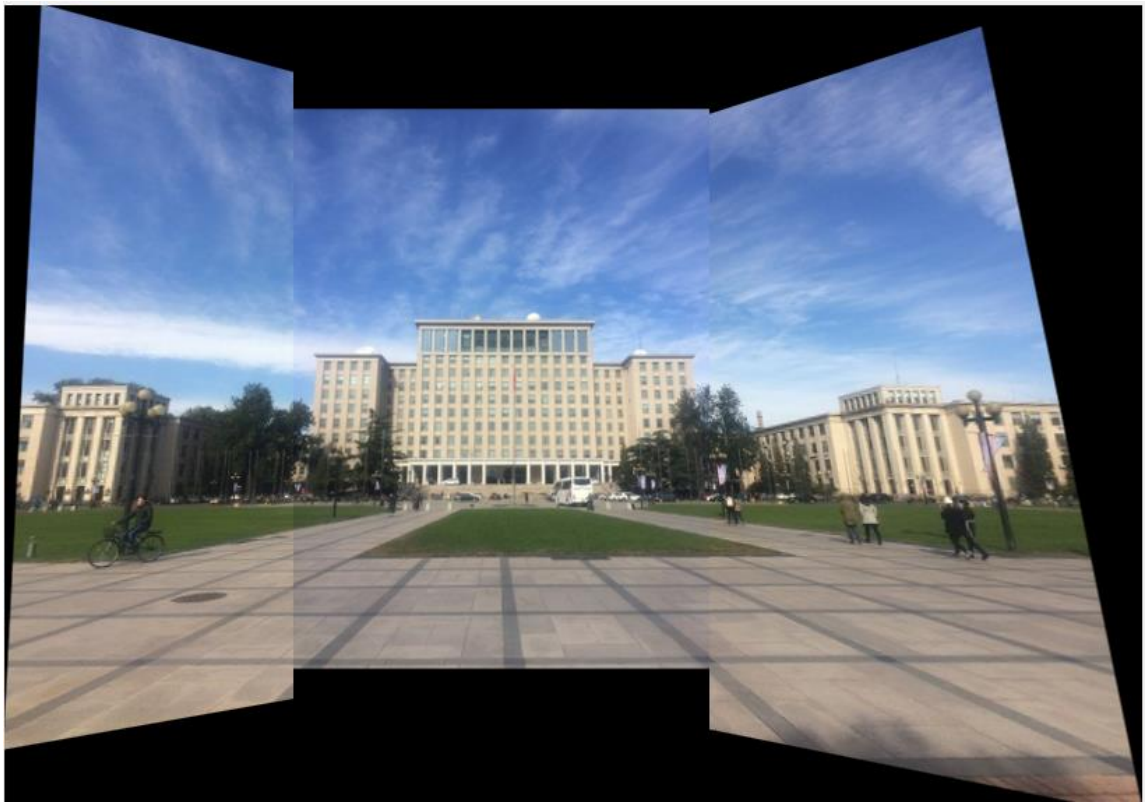
依次放上 A’，B’，C’ 的效果图如下：

放上 A’的一部分，

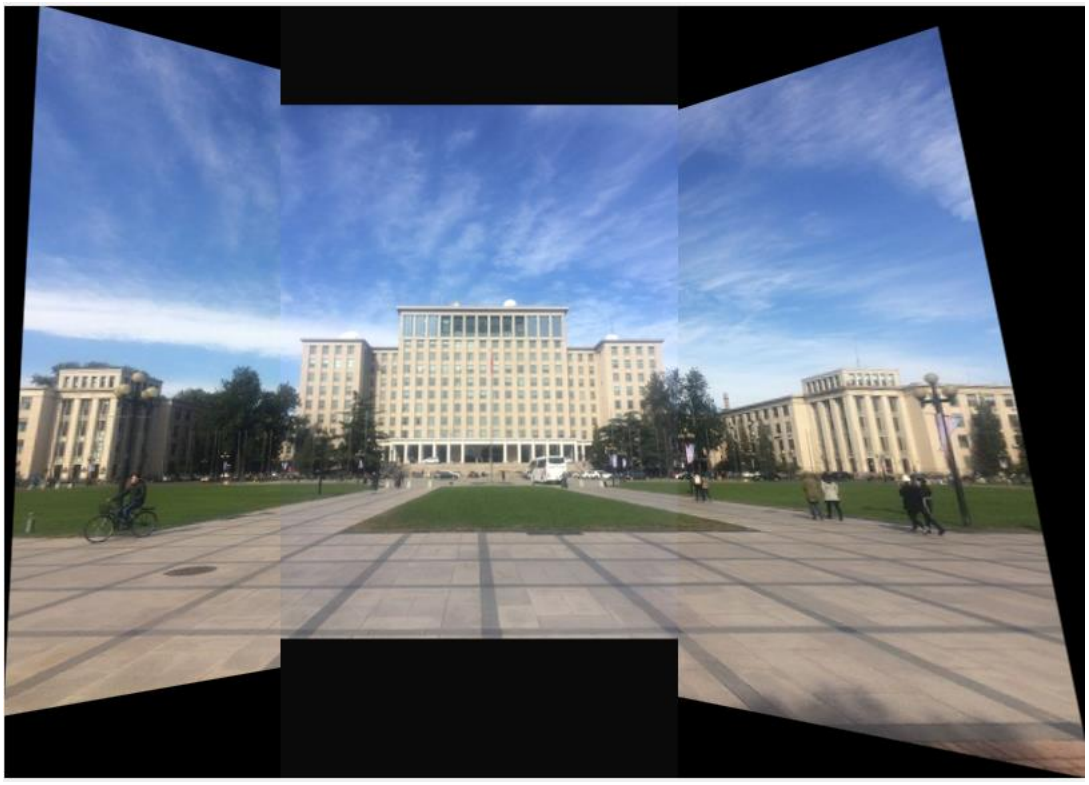


再放上 B’的全部，

最后放上 C’的一部分，



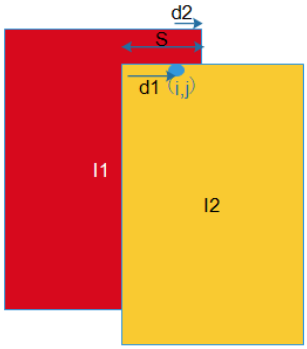
由于中间一幅图的曝光较暗，对其亮度加成一定值之后，效果如下



区分度稍微减小了一些，下面再对它进行渐变处理。

2.6 渐变的图像融合算法

在和同学(邱楚聿)讨论之后，得知可以对每两幅图像的重合部分进行渐变处理，这样可以减弱图像之间的接缝，原理和上面的融合算法类似，不过边界条件进行了改变，并且引入了加权值。这个加权值和当前像素的横坐标具边界的距离有关。如下图所示，



对(i,j)点的像素值 p(i,j)，有

$$p(i,j) = \frac{d1}{s} \times I_1(i,j) + \frac{d2}{s} \times I_2(i,j) \qquad (1)$$

具体算法如下：

算法 2-3 渐变的图像融合算法
输入：透视变换矩阵 T1，T2(单位阵)，T3，图片 A，B，C 输出：全景图 D 1. 分别计算矩阵 A，B，C 在透视变换下的 x (y 方向同理，这里不再赘述) 方向坐标的最大值和最小值，分别记为[x_A_low, x_A_high],[x_B_low, x_B_high],[x_C_low, x_C_high]； 2. 选取 x_lowest=min{ x_A_low , x_B_low, x_C_low }; 选取 x_highest=max{ x_A_high , x_B_high, x_C_high }; 3. 生成空的全景图，全景图的宽度为 x_highest-x_lowest；(高度为 y_highest-y_lowest) 4. 在全景图上，宽度从 1 到 x_B_low - x_A_low 处，放置 A 的投影图； 宽度从 x_B_low - x_A_low 到 x_A_high - x_A_low 处，是 A、B 的重叠处，用公式(1)计算像素值； 宽度从 x_A_high- x_A_low 到 x_C_low,- x_A_low 处，放置 B 的投影图； 宽度从 x_C_low,- x_A_low 到 x_B_high - x_A_low 处，是 B、C 的重叠处，用公式(1)计算像素值； 宽度从 x_B_high - x_A_low 到 x_C_high - x_A_low 处，放置 C 的投影图； 5. 输出全景图；

2.7 渐变的图像融合算法的效果

采用了渐变算法之后，图像的效果有了很大的提升，如下，

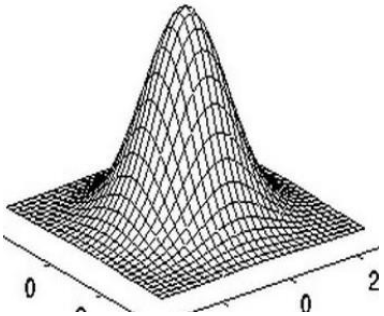
可以看到，箭头所示图像的接缝处得到了很好的过渡，但缺点是主楼变得模糊，因为 A、C 变换之后的主楼无法完美地和 B 图重合在一起。

2.8 频域低通滤波算法

这个算法十分简单，只要按照普通的高斯低通滤波算法即可，具体如下

算法 2-4 频域低通滤波算法
输入：全景图 D，sigma 输出：滤波后的全景图 D’ 1. 根据 sigma 创建高斯函数 mask； 2. 把 D 移到中心为 0 的坐标系，对其做快速傅里叶变换； 3. Out=fft(D)*mask； 4. 把 Out 移到中心，对其做快速反傅里叶变换； 5. 输出 Out；

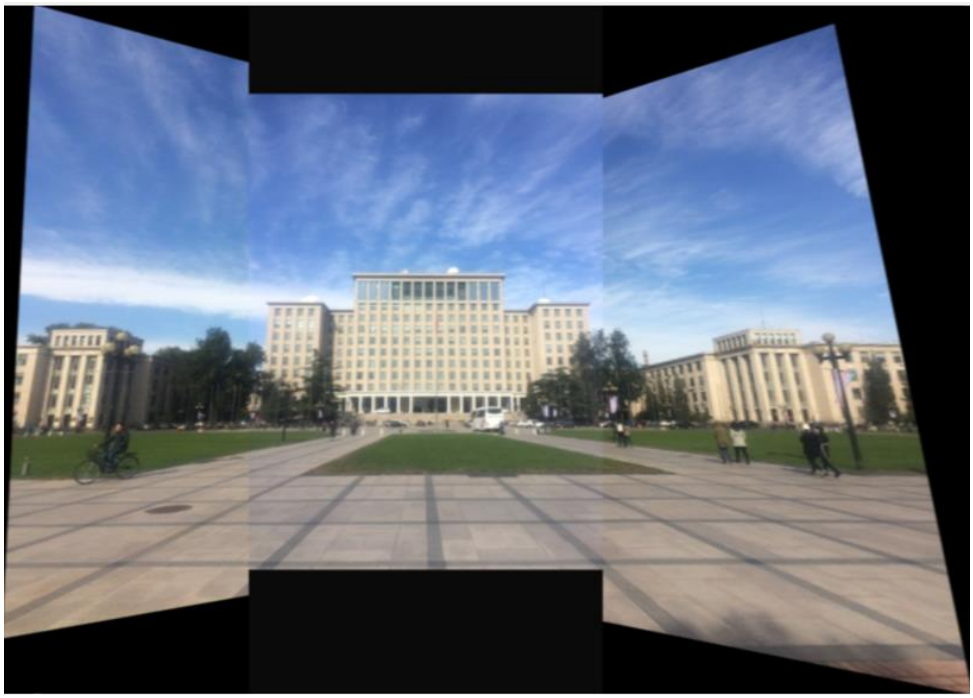
由于高斯函数如下图所示



所以图像的高频部分被削减，图像变得平滑，理论上接缝处明显度会降低；

2.9 频域低通滤波的效果

结果如下：



可以看到，左侧接缝不那么明显了，而右侧接缝依旧较明显，滤波后图像清晰度也变低。

2.10 本章小结

手动兴趣点匹配的拼接算法主要由透视变换、拼接、滤波三部分组成，全部的算法和编程都由自己实现。由于特征点是手选的，变换矩阵通过奇异值分解计算出来；拼接算法思路清晰，并且用渐进减弱了接缝；滤波算法是经典的高斯低通滤波器；所以本部分的算法总体不是那么复杂，计算速度也非常快，但缺点是只能用在特定的三幅图中。

第三章　基于 Harris 角点匹配的拼接算法

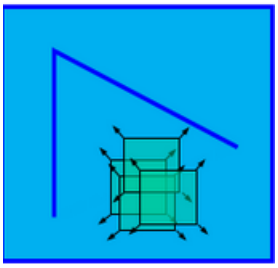
3.1 理论分析

由于手动选点的方法没有通用性，且有主观因素的影响。下面的想法是，希望找到一个基于数学模型的，能自动挑选出图片中特征的算法。Harris 角点算法应运而生。

“角点”顾名思义，可视为所有平面的交汇处或发起处，若改变了这个点的位置，其交汇的平面必定会发生很大的改变。所以，这便引出了图像角点的定义——若某一点在任意方向的一个微小变化都会引发灰度值很大的变化，那么这个点就是角点。

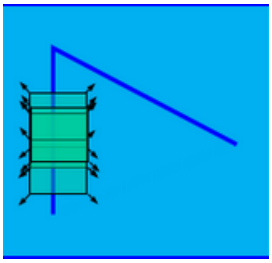
下面我们对其进行数学上的定义。

当一个窗口在图像上移动，若窗口在各个方向上没有变化，如下图所示，



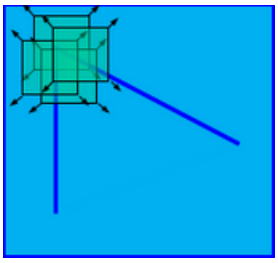
说明其变化率为 0，即一阶导数或者梯度为 0；

而若窗口在某条直线或者图像的边缘移动，其沿直线或边缘方向没有变化，如下图所示，



说明其在某一方向的一阶导数为 0。

当窗口在角点处，窗口在各个方向均有变化，若下图所示，



所以用窗口在各个方向上的变化程度来衡量一个点是否为角点。

将图像窗口平移[u,v]产生灰度变化 E(u,v)

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

由：$I(x+u,y+v)=I(x,y)+I_x u+I_y v+O(u^2,v^2)$，得到：

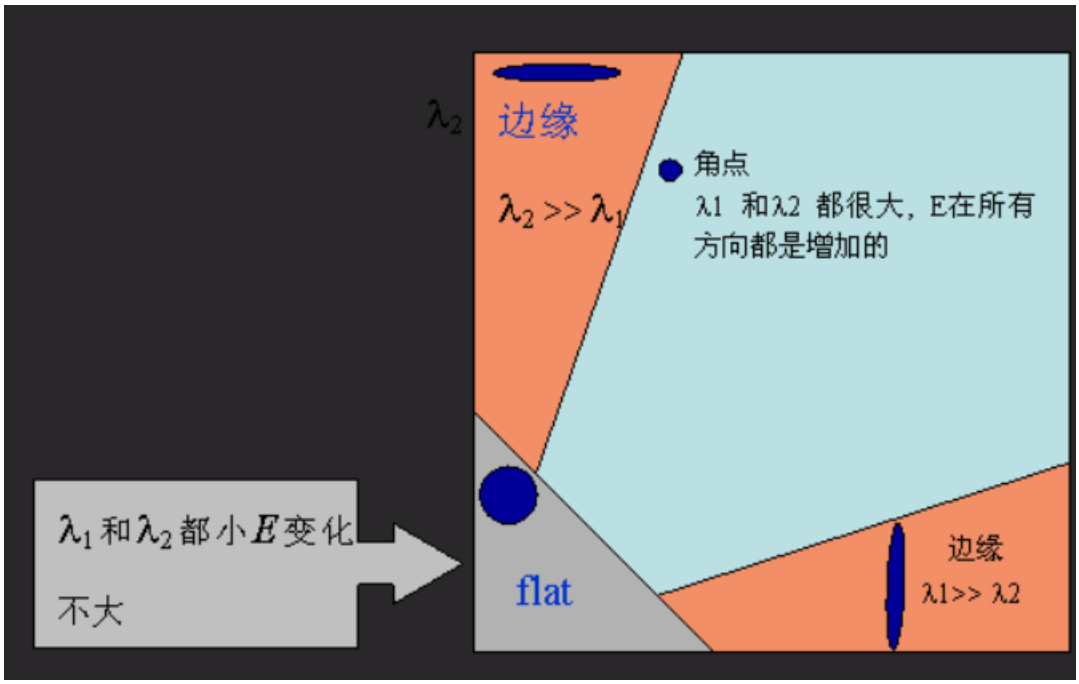
$$E(u,v)=[u,v]\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

对于局部微小的移动量 [u,v]，近似表达为：

$$E(u,v)\cong[u,v]M\begin{bmatrix} u \\ v \end{bmatrix}$$

其中 M 是 2*2 矩阵，可由图像的导数求得：

$$M=\sum_{x,y} w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$



为了判断角点的质量，定义响应函数为：

$$R=\lambda_1\lambda_2-k\left(\lambda_1+\lambda_2\right)^2$$

但解特征值和特征向量的计算复杂度很大，由线性代数的结论：两个特征值的和等于矩阵 M 的迹，两个特征值的积等于矩阵 M 的行列式。

所以 R 转为下面的式子：

$$R = \det M - K(\mathrm{trace}M)^2$$

其中 K 是经验参数，在(0.04,0.06)之间取值，这里我们取 K=0.06；

通过一个阈值，即可规定

$$R > \mathrm{threshold}$$

的点为角点。这里threshold取 R 中最大值的某个倍数，确定值通过不断尝试之后取个较好的值；

在找出每副图的角点之后，还需要计算机代替人来对两幅图中的角点进行匹配，这里用到了经典的归一化交叉相关相似性度量函数(Normalized cross correlation， NCC)算法，来匹配两个窗口间的相似程度。下面简单分析 NCC 的原理。

NCC 匹配算法[6]是一种经典的匹配算法。通过计算模板图像和背景图像的互相关值确定匹配的程度。互相关的最大值决定了模板图像在背景图像中的位置和相似程度。

在实际匹配应用中，背景图和模板图的相似性通过度量函数来度量，则归一化积相关匹配度量定义为：

$$r(i,j)=\frac{\sum_m\sum_n\left(A_{mn}-\bar{A}\right)\left(B_{mn}-\bar{B}\right)}{\sqrt{\sum_m\sum_n\left(\left(A_{mn}-\bar{A}\right)^2\right)\sum_m\sum_n\left(\left(B_{mn}-\bar{B}\right)^2\right)}}$$

式中：

r(i, j)为子图左上角顶点在背景图中的坐标；

A 为背景图像；

B 为模板图像；

m 和 n 分别为图像的行和列。

NCC 算法具有很高的准确性与适应性，对图像灰度值线性变换具有“免疫性”，，即所求的 NCC 值不受灰度值的线性变换的影响。

于是在角点匹配中，可以生成 A 图某个角点周围的一定大小的窗口，遍历 B 图所有角点的窗，找出最相似的两个窗口。这样，就实现了角点的匹配。

3.2 Harris 角点探测算法

把上面角点探测的思路编程，注意角点探测是针对灰度图的，并且需要对浮点数操作。算法具体如下：

算法 3-1 Harris 角点探测算法

[6]吴 忠.一种简化、高效的 NCC 图像匹配算法.[J]科技传播，2013

输入：图片 I1，threshold

输出：角点储存数组 Result，角点数 cnt

1. RGB 图像灰度化，生成梯度算子，计算每个点 x,y 方向的梯度 Ix，Iy；
2. 计算梯度方向的乘积

$$Ix^2 = Ix \times Ix;$$

$$Iy^2 = Ix \times Iy;$$

$$Ixy = Ix \times Iy;$$

3. 使用高斯函数对Ix^2, Iy^2, Ixy进行加权，计算 detM 和 traceM；

$$detM = Ix^2 \times Iy^2 - Ixy \times Ixy$$

$$traceM = Ix^2 + Iy^2$$

4. 计算角点量 R，若 R> $threshold \times \max(R(,))$，则认为其是备选角点；
5. 对备选角点进行局部非极大值抑制：

   若$R(i,j)$ >3*3 窗口内的所有邻居，则认为其为真正角点，Result(i,j)=1,cnt++；

   否则，抑制，$R(i,j)$不是角点；
6. 输出 Result 和 cnt；

最后，遍历 Result(i,j)，寻找值为 1 的点，在对应的 RGB 图的相应位置标明角点即可。

3.3 Harris 角点探测算法的效果

对三幅图片分别进行 Harris 角点探测，并标记，结果如下，

图 A 的所有 Harris 角点

图 B 的所有 Harris 角点

图 C 的所有 Harris 角点

可见，从肉眼上看，图中角点标明的部位基本都符合其“角”点的定义。

3.4 Harris 角点匹配算法

把关于 NCC 匹配的思路编程，这里固定窗口大小为 11*11，首先要实现窗口的归一化交叉相关相似度的计算，具体算法如下：

算法 3-2 归一化交叉相关相似度的算法
输入：窗口 A, B 输出：归一化交叉相关相似度 ncc 1. 计算$\mathrm{N1} = A - \bar{A}$，$\mathrm{N2} = B - \bar{B}$ 2. 计算 $$\mathrm{ncc} = \frac{sum(\mathrm{N1} \times \mathrm{N2})}{\sqrt{sum(\mathrm{N1}^2) \times sum(\mathrm{N2}^2)}}$$ 这里的 sum 表示矩阵所有元素相加。 3. 输出 ncc。

接着，根据 ncc 进行匹配，处理时注意图像的边界，这里的思路是把图像宽和高各扩大 5 个像素，具体算法如下：

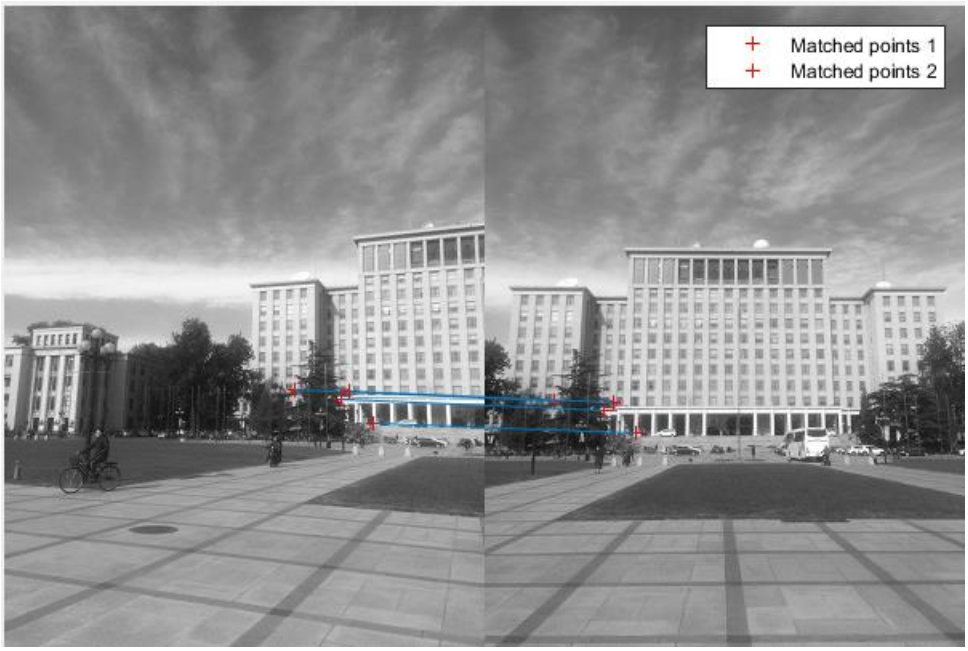
算法 3-3 Harris 角点匹配算法
输入： I1_gray：图片 I1； I2_gray：图片 I2； I1_Harris：图片 I1 的角点矩阵； I2_Harris：图片 I2 的角点矩阵； I1_cnt：I1 的角点数； I2_cnt：I2 的角点数； 输出：match_pt1：I1 中匹配角点的坐标； match_pt2：I2 中匹配角点的坐标； 1. 取出 I1、I2 的 Harris 角点； I1、I2 宽和高各扩大 5 个像素； 2. 遍历 I1 的角点，设当前角点为(x,y)，建立(x,y)周围11×11的窗 A； 3. 针对窗 A，遍历 I2 的角点，建立11×11的窗 B，计算所有(B，A)的 ncc，若这个 ncc 除以最大 ncc 小于某个阈值，则这个窗对应的角点(x',y')匹配于 A 的角点(x,y)； 4. 记录这上述匹配点的坐标，继续遍历 I1 的角点； 5. 输出匹配点坐标矩阵。

注意 NCC 算法也是建立在灰度图像上的；

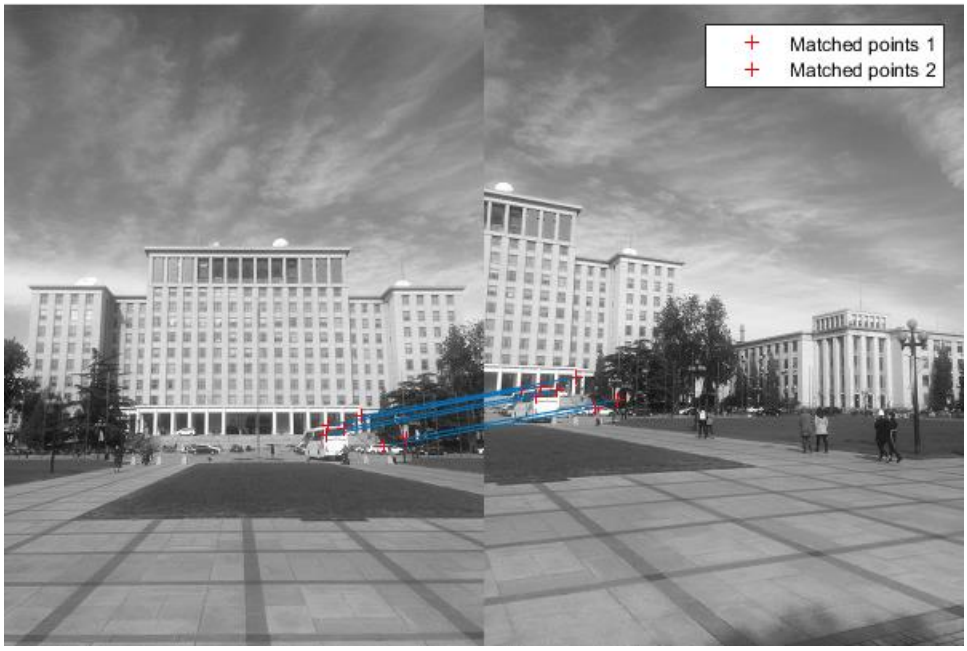
3.5 Harris 角点匹配算法的效果

为了方便演示，这里还实现了一个绘画算法，即把两幅图拼在一起输出，同时在图上对应点之间画一条线即可，这个算法比较简单，在此不赘述。标记的结果如下：

A 与 B 的角点匹配



C 与 B 的角点匹配

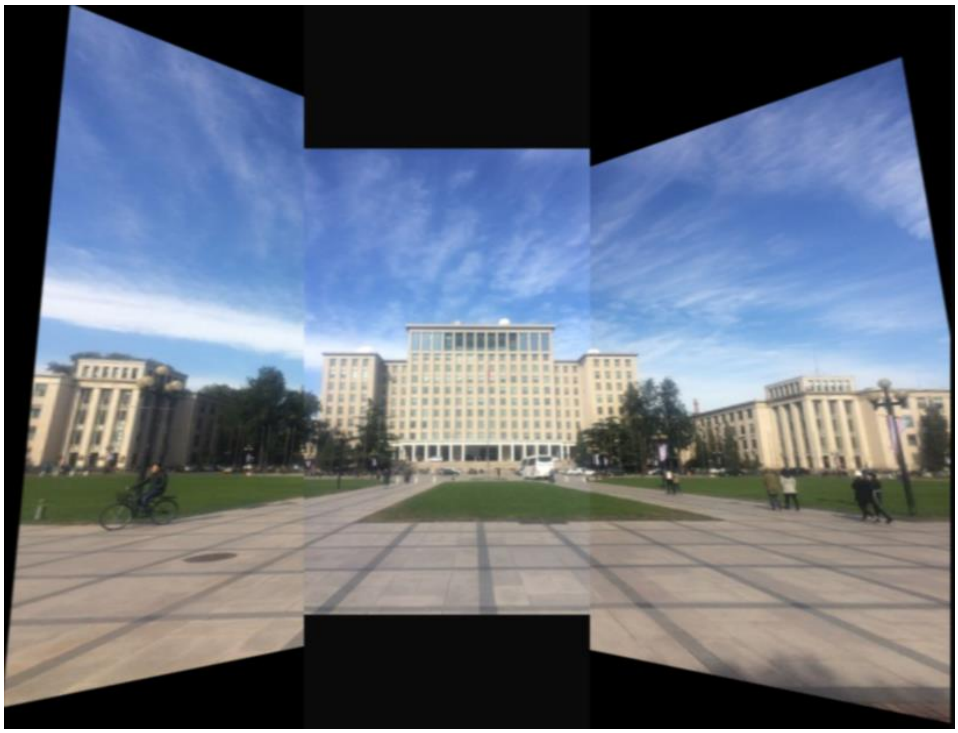


可以看到，虽然匹配的点很少，但基本没有误匹配。说明自己实现的算法是正确的！

3.6 图像融合算法及效果

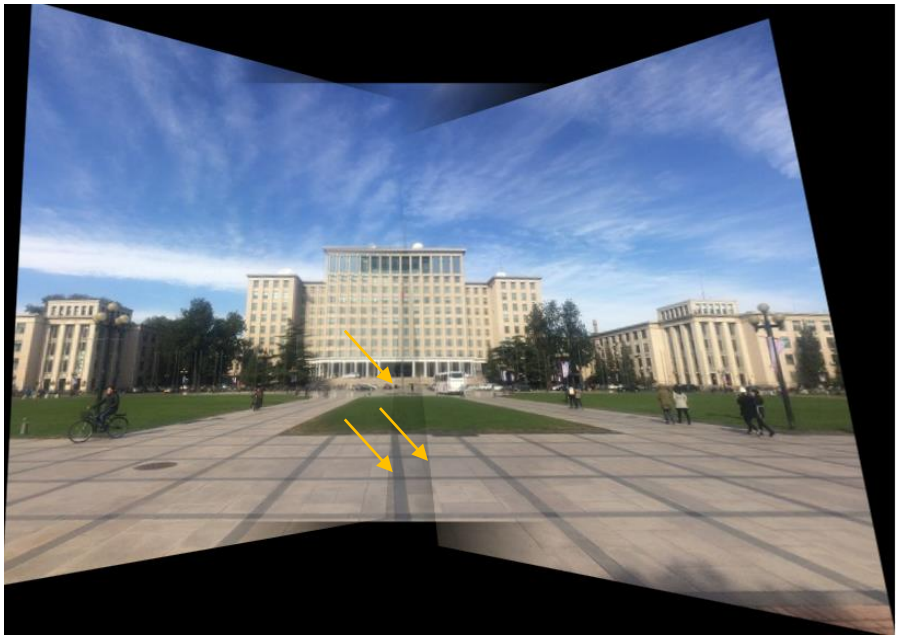
既然 Harris 角点匹配算法已经找到两幅图中的对应点，且对应点大于等于 4 个，那么只需要把在对应点中取 4 个出来，放到第二章中已经实现的融合算法中，算出对应透视变换矩阵，分别变换并输出即可。最后依旧对图像经行频域低通滤波处理。

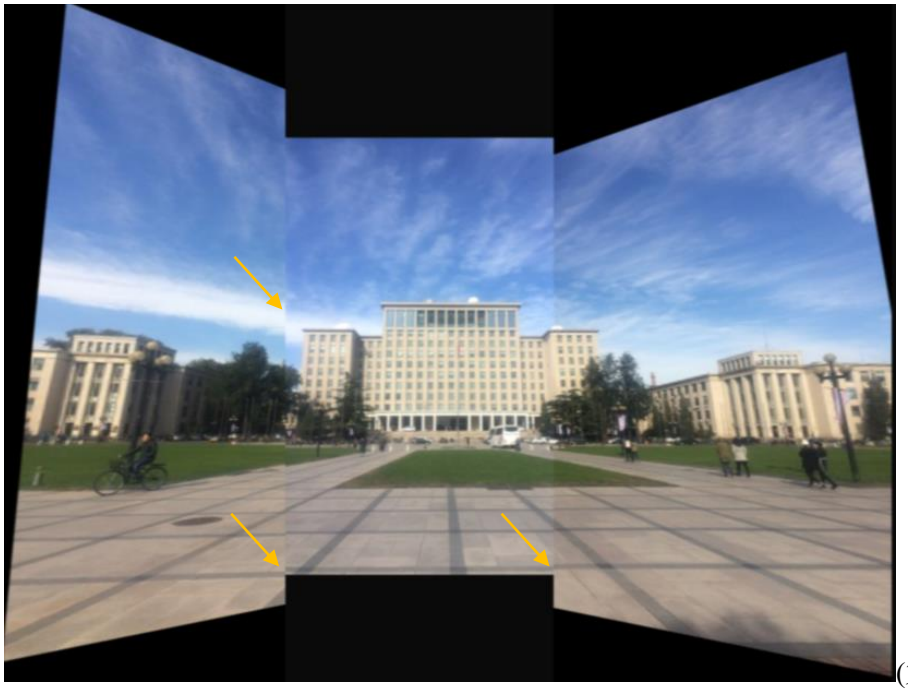
最后的效果如下所示



3.7 手动算法和 Harris 角点算法的比较

这两种算法均是自己编程实现的，没有调用任何相关的库函数，下面对两幅图进行对比：


(第一种算法)


(第二种算法)

仔细观察发现，第一种算法拼出的“地板”衔接比较好，而第二种算法的“地板”出现了错位，同时第一种算法的云彩的拼接也优于第二种算法。但第二种算法理论上可以适用于其他的图片，但第一种算法需要每次都手动选点。

3.8 本章小结

本章利用 Harris 角点的梯度定义，自行设计并实现了 Harris 角点的侦查算法，结合归一化交叉相关相似度的定义，自行设计并实现了 Harris 角点的匹配算法，最后利用匹配的点代入到第二章的透射变换矩阵算法和融合算法中，实现了自动的图像融合。最后的结果表明此方法效果略有瑕疵，但整体还是非常可行的。

第四章　基于 Surf 特征点匹配的拼接算法

4.1 理论分析

在第一章的背景叙述中，提到了 Sift 算法，而 Surf 算法正是尺度不变特征变换算法（Sift 算法）的加速版，在执行效率和鲁棒性上更胜一筹。

具体的 surf 算法原理[7]和 Hessian 矩阵、金字塔尺度空间和特征点描述子有关，由于本课题中调用了 Matlab 的库函数，为此不再对具体的实现过程进行理论证明。

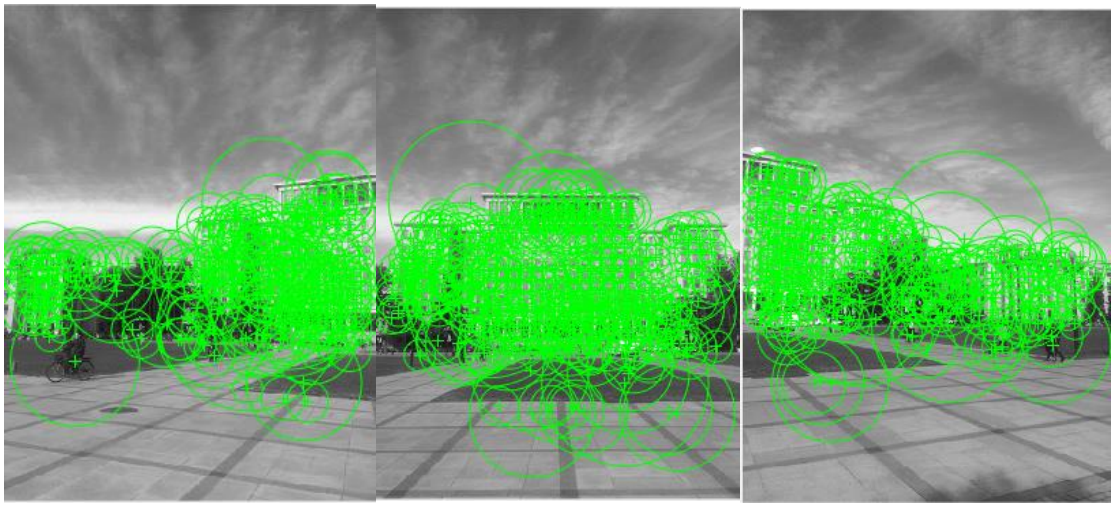
4.2 基于 Surf 特征点匹配的拼接算法及效果

下面借助 Matlab，分模块叙述 Surf 特征点匹配的拼接算法。

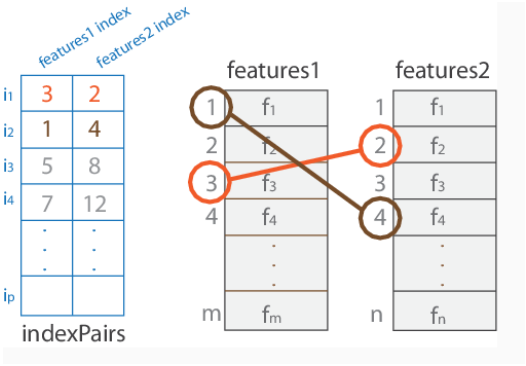
算法 4-1 基于 Surf 特征点的特征寻找
输入：图片 I1； 输出：特征点 valid_points，特征矩阵 valid_features； 1. 读取 I1，转化为灰度图； 2. 调用 detectSURFFeatures 函数，输出 SURFPoints 类的备选特征点； 3. 调用 SURFPoints 类已封装的 plot 函数，画出特征点； 4. 调用 extractFeatures 函数，取回有效的 valid_points 和周围的特征矩阵 valid_features； 5. 输出 SURFPoints 类的特征点和特征矩阵

Surf 特征点的特征寻找效果非常好，如下图所示，

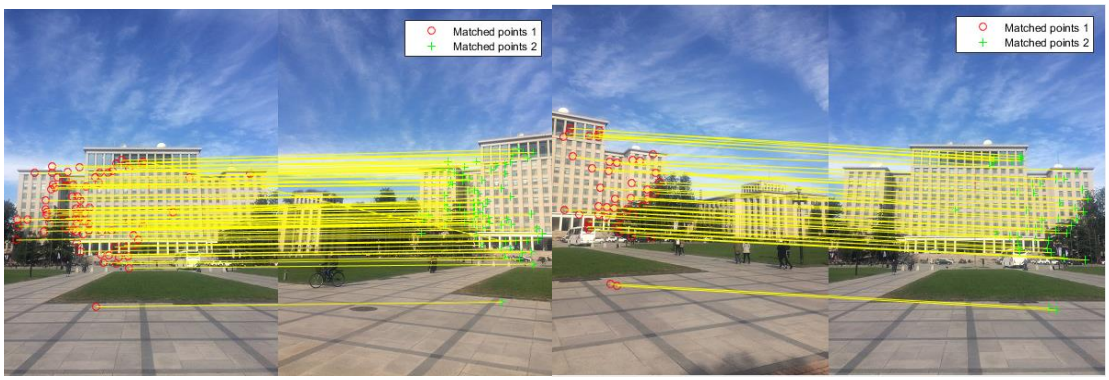
[7] http://www.360doc.com/content/11/1129/15/3054335_168366315.shtml



找到特征点之后，要进行特征点匹配，算法如下，

算法 4-2 基于 Surf 特征点的特征匹配
输入：图片 I1 的特征矩阵 valid_features1，图片 I2 的特征矩阵 valid_features2； 输出：特征索引矩阵 index 1. 调用 matchFeatures 函数，生成特征点索引，示意图如下：    通过索引可以很方便用查表的方法知道匹配的点。输出索引 Index。 2. 调用 showMatchedFeatures 函数，利用索引，输出匹配点；

效果如下，



观察图片可知，surf 特征点匹配的很好，几乎没有误匹配，并且多数特征点集中在主楼上，这符合我们的期望。

接下来，我们利用这些匹配点进行透视变换，这里注意，此 surf 算法可以默认图片顺序任意，所以一开始的透视变换均是对于 I1 做出的，而最后需要找寻位于中间的图片，再针对此图片改变透视变换。计算透视变换时用到 projected2d 类投影变换数组，它封装了3×3的透视矩阵和维度信息。

算法 4-3 基于 Surf 特征点的透视变换
输入：图片 I1 的特征点 valid_points1，图片 I2 的特征点 valid_points2，图片 I3 的特征点 valid_points3 输出：projected2d 类数组 tform(3)； 1. 调用 estimateGeometricTransform 函数，分别输入特征点，输出相对于 I1 的 tform 矩阵； 2. 调用 outputLimits 函数，分别每幅图计算透视变换后的 x、y 限制； 3. 根据这些限制，找到位于中间的图片 I2， $$\text{tform(i)} = \text{tform(2)}^{-1} \times \text{tform(i)}$$ 4. 输出 tform(3)；

有了变换矩阵，最后就是根据这个矩阵进行全景图的生成了，算法如下，

算法 4-4 基于 Surf 特征点的全景图生成
输入： I1,I2,I3：三幅 RGB 图片， tforms：对应的 tforms 对象数组 x_lim：x 方向的限制， y_lim：y 方向的限制

输出：panorama_pic：全景图

1. 根据变换之后最大最小的坐标限制，创建空的全景图；
2. 调用 imref2d 函数，创建世界坐标系的视角 perspective；
3. 调用 iwarp 函数，在 perspective 视角下，生成一个二值的 mask；
4. 调用 step 函数，利用 mask 和 tform，在 perspective 视角下进行透视变换；
5. 输出全景图 panorama_pic；

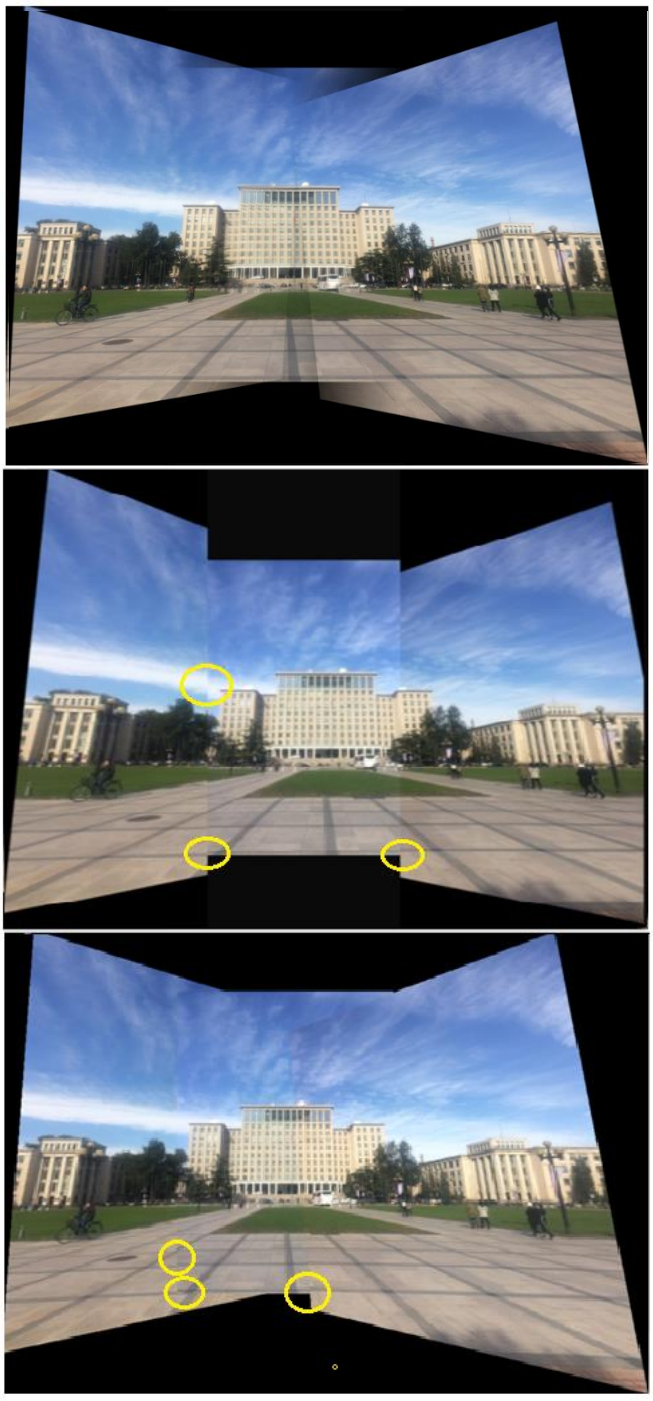
同样地，滤波之后，结果如下，

结果表明，此算法运行良好，下一章将对三种方法进行对比讨论。在此就不再总结赘述。

第五章　三种算法的优劣对比

5.1 效果对比

下面对比三种算法的效果，如下图



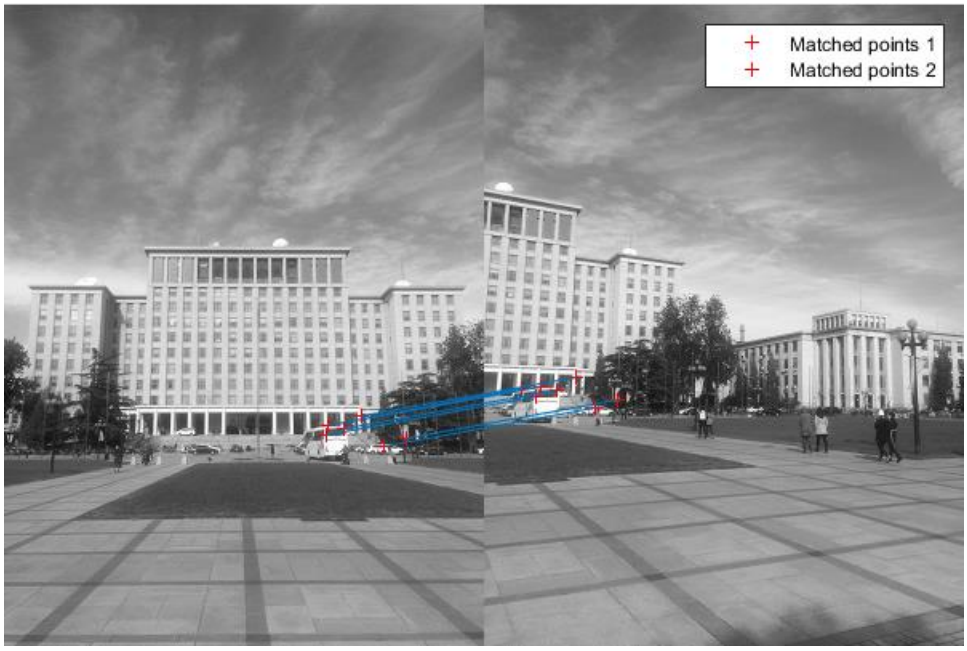
黄色圆圈圈出的地方，是相比于第一幅图中衔接的不好的地方，可见手动选点在匹配度上高于后两种算法。

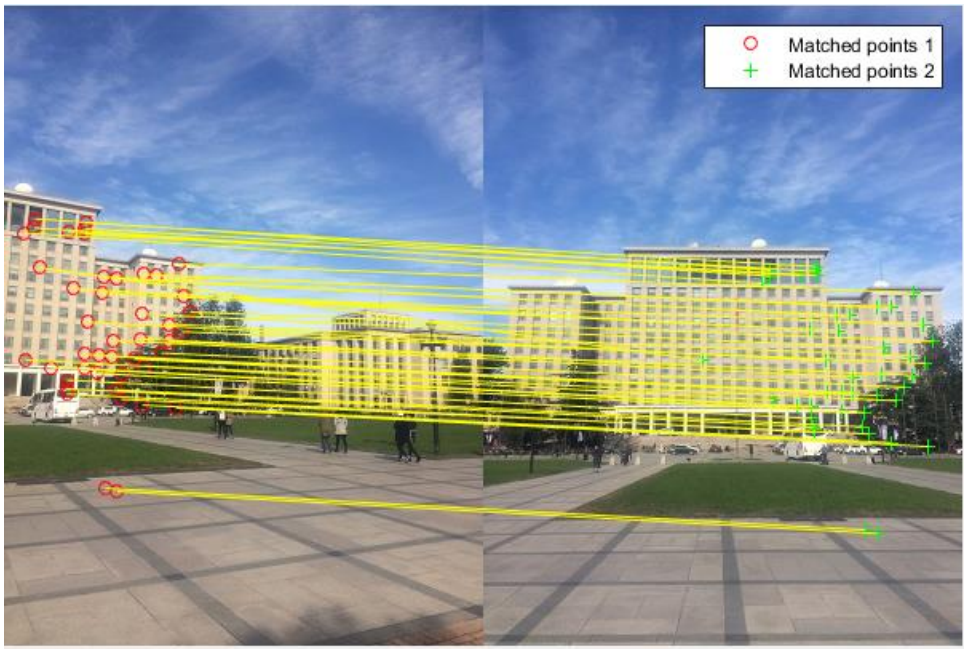
另一方面，第一种算法用了渐变的融合方法，而最后一种调用了 step 函数，所以两者接缝看起来没有第二种那么明显。

5.2 算法复杂度对比

从算法复杂度来看，手动选点的方法简单了不少，因为它跳过了比较复杂和容易出错的特征点匹配，而把这项工作交给了人，缺点是无法适用在其他图片上。

后两种方法都是基于自动选取特征点的。其中 Harris 角点匹配法由自己实现，虽然能用几乎 100%的正确率够匹配出特征点，但匹配的点较少；Surf 算法使用了库函数，能匹配出很多的特征点，正确率也比较高。两种算法的特征点匹配如下所示：





5.3 算法速度对比

从速度上来看，前两种算法都比较快，最后一种算法由于匹配点很多，速度有些慢。

5.4 对比结论

下面列表总结三种算法的比价，

	算法效果	算法复杂度	算法速度	实现方法
手动兴趣点匹配	地板拼接优秀； 但接缝较明显；	中等	快	所有函数自己编程实现
Harris 角点匹配	地板拼接优有瑕疵； 接缝较明显；	较难	快	所有函数自己编程实现
Surf 特征点匹配	地板拼接有瑕疵； 接缝较不明显；	难	较慢	使用库函数编程实现

第六章　总结与展望

本文基于特征点的匹配的思想，用手动兴趣点匹配、Harris 角点匹配、Surf 特征点匹配三种方法，实现了三幅数字图像的拼接。其中，手动兴趣点匹配、Harris 角点匹配全部函数的编程工作由自己完成，Surf特征点匹配的方法借用了 Matlab 的库函数工具包。本文整理、分析了三种方法的原理，并且对其性能以及效果进行了分析和比较。此外，本文利用频域低通滤波器，对拼接后的图像进行了平滑处理。

由于 Harris 算法的结果不很理想，未来的工作将着眼于优化其匹配算法上，可行的方法是利用 RANSA 算法去除误匹配点。

参考文献

1. C.Kuglin, D.Hines. The Phase Correlation Image Alignment Method[C]IEEE Conf. 1975
2. Deriche R. Recovering and characterizing image features using an efficient model based approach.[C] IEEE Computer Vision Conference. 1994
3. 谭康. 图像拼接技术研究与实现[D]南京,2006
4. 张茂军，虚拟现实系统[M]北京：科学出版社，2010
5. 吴 忠.一种简化、高效的 NCC 图像匹配算法.[J]科技传播，2013
6. http://blog.csdn.net/xiaowei_cqu/article/details/26471527
7. http://www.360doc.com/content/11/1129/15/3054335_168366315.shtml