数字图像处理作业 综合作业 1

图像拼接

项目报告

晏筱雯

自 42

2014011459

2016.11.02

目录

一、作业要求

请各位同学采取适当方法，对附件中三张照片进行图像拼接，得到一副完整图片。

二、方法原理

这次作业要求拼接三张拍摄位置、角度、时间等均不同的照片，为了让拼接的结果比较合理，需要首先对图片进行仿射变换，使三张图片的视角基本一致再进行拼接；对于照片中重合的部分，通过给两张图片的像素值以适当的权重使两张图片较为平滑地融合在一起而没有明显的分界。

三、实验思路和算法设计

1.仿射变换

仿射变换仿射变换（Affine Transformation 或 Affine Map）是一种二维坐标到二维坐标之间的线性变换，它保持了二维图形的“平直性”（即：直线经过变换之后依然是直线）和“平行性”（即：二维图形之间的相对位置关系保持不变，平行线依然是平行线，且直线上点的位置顺序不变）。仿射变换包含了平移、旋转、错切、翻转、缩放等。

于是，这次的大作业中，我通过选取特征点、求仿射变换矩阵、对原图像做变换来完成图像的初步处理。

（1） 选取特征点

Matlab 自带的函数 sift 是目前常用的 local feature 的描述子。sift 特征匹配算法可以处理两幅图像之间发生一些平移、旋转、仿射等匹配问题。但是考虑到 sift 函数选取的特征点过多不便处理，（并且也不允许用），故我选择用 MATLAB 中的 cpselect 函数手动选取特征点的坐标。

如图 1 和图 2 所示，由于第二张图（中主）比较正，且位于图像的正中央，因此在选取特征点时，我以第二张图作为基准。

Cpselect 函数的输出值是一个$n \times 2$的矩阵，第 1、2 列分别为特征点的横纵坐标。图 1 中我选取了 4 对特征点，但是实际上只用到了 1~3 组。输出 2 组共 4 个矩阵之后，我用 save 命令把矩阵保存在 keypoints.mat 文件中，在后续的程序中直接使用 load 命令调用即可。

（2） 求仿射变换矩阵

仿射变换的变换矩阵 T 可以用一个3×3的矩阵表示：

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

设原来的点的坐标为(x, y)并补成$(x, y, 1)$的形式，仿射变换后的点的坐标为

(x', y')，那么我们可以得到如下方程：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

如果找到三对这样的点，我们就可以得到如下的方程组：

$$\begin{bmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}$$

易得：

$$g = h = 0, i = 1$$

令$P = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}, P' = \begin{bmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1 \end{bmatrix}$，则

$$T = P' \times P^{-1}$$

而 MATLAB 中的函数 pinv 可以很快地求出矩阵P的逆矩阵，从而求解出变换矩阵 T。

（3） 对原图像做变换

对新坐标求逆仿射变换后即可得到其对应在原图上的点的坐标，这个坐标很有可能不是正整数，此时需要对原坐标进行插值，将对应点的像素值赋给新的坐标。对于插值方法，这里我采用了最近邻插值法。

最近邻插值法是最简单的灰度值插值。也称作零阶插值，就是令变换后像素的灰度值等于距它最近的输入像素的灰度值。即对于变换后的点$f(x', y')$，利用仿射变换矩阵的逆矩阵求得其在原图中的对应点$f(x, y)$，由于x, y可能不是整数，因此我利用下面的公式赋值：

$$f(x', y') = f(floor(x + 0.5), floor(y + 0.5))$$

（4） 实际采用的方法

对于上述求仿射矩阵和做仿射变换的方法，我在刚开始的几次尝试中发现由于程序中有如下代码：

```
for i=1:m
    new(1,(n*(i-1)+1):n*i)=i;
    new(2,(n*(i-1)+1):n*i)=1:n;
    new(3,(n*(i-1)+1):n*i)=1;
end
```

即创建一个$3\times mn$的矩阵，而m和n都是四位数，这使得程序的速度非常缓慢。

我了解到 MATLAB 本身有进行仿射变换的函数 maketform 和 imtransform，通过 maketform 函数得到一个结构体，这个结构体中包含着仿射变换需要的矩阵等信息；这个结构体即为 imtransform 函数的输入值之一，再指定需要仿射变换的函数，即可得到相应的图形。

改用了这两个函数之后，我的程序果然变快了很多，因此程序中保留了这一方法。

2.图像拼接与融合

（1）图像拼接

由于三张图片中有重合的部分，因此无法简单直接地把它们拼接在一起；如果把重合的部分从某张图片上截去，又不符合图像拼接的实质，因此我首先要找到重合的区域。这里我还是采用了 cpselect 手动选取点的方法，得到了边界点在对应仿射后的图片上的位置，从而确定了重合区域的宽度d_1（第一张与第二张图片的重合宽度）和d_2（第二张与第三张图片的重合宽度）。于是拼接后新图片的宽度为

$$d = d_{image1} + d_{image2} + d_{image3} - d_1 - d_2$$

而拼接后图像的高度由于拼接位置的差异无法根据原来的图像得到，因此我给了一个尽量高的高度以避免新图像溢出。

（2）图像融合

三张图片拍摄的位置不同、时间不同，这使得它们的亮度饱和度等参数有着肉眼可见的差异，因此对于拼接后的图片的重合区域，需要通过加权的方式使得色调、亮度等平滑过渡。加权的方式有很多种，基本的原则就是如果重合区域中的某个点离左边的图更近，那么左边的图的权重就应该更大，反之右边的图

的权重更大。因此可以设重合区域中的某个点离重合区域左边界距离为d_1，离右边界距离为d_2，由下面的公式：

$$f = \frac{d_2}{d_1+d_2} \cdot f_{left} + \frac{d_1}{d_1+d_2} \cdot f_{right}$$

计算出该点的像素值。

四、结果讨论

1. 选取特征点

特征点选取结果如下：

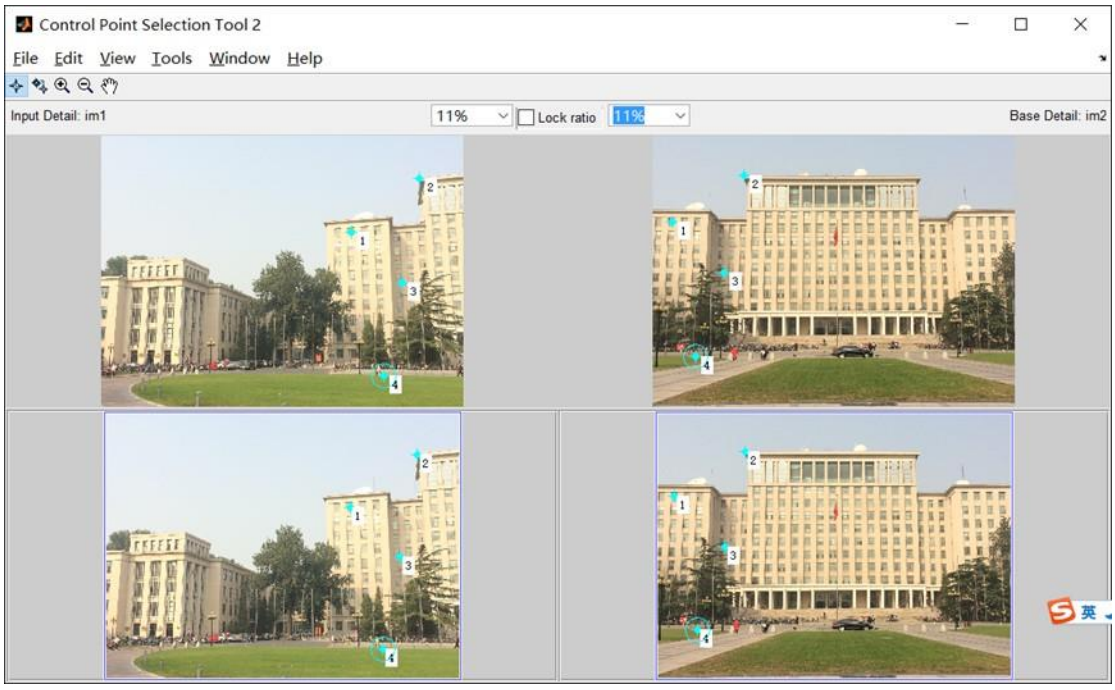

图 1：第一张图和第二张图特征点选取结果

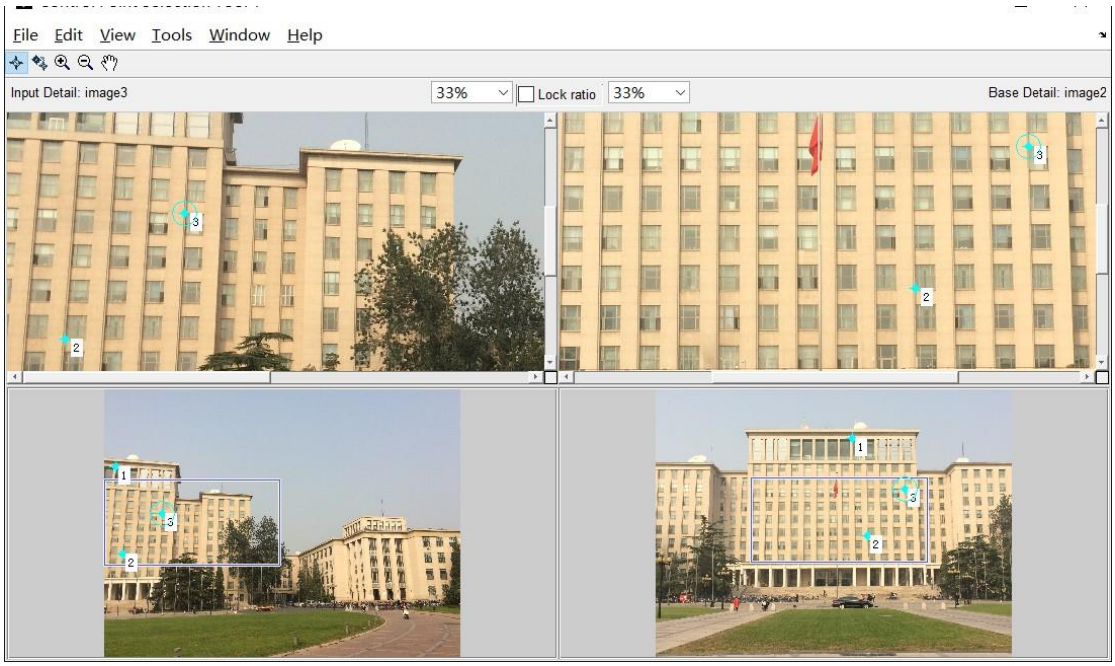

图 2：第二张图和第三张图特征点选取结果

2.仿射变换

对第一张图片（西主）和第三张图片（东主）仿射变换后如下图 3 和图 4：

图 3 对第一张图片（西主）仿射变换的结果

图 4:第三张图片（东主）仿射变换的结果

可以看到，仿射变换后的图片的中主部分基本能与地面垂直了，这说明仿射的目的达到了。

另外，根据后面拼接时的尝试，由于我们希望拼接的结果是重合的部分能够融合得比较好，因此在选取特征点时，应尽量选取重合部分的点，并且选取的点应尽可能平均分布在重合区域。

3.拼接与融合

最终的拼接和融合结果如下：

图 5 图像拼接结果

从小图来看，此次图像拼接的整体效果还可以，主楼的主体轮廓基本不失真，天空的颜色平滑过渡。但是放大了之后发现重合区域窗户和树的重影较多，地面上的两条道路都没有了。

分析原因如下：

① 仿射变换不够细致。MATLAB 自带的仿射函数要求输入的是 3×2 的矩阵，即只有三对特征点，对于一张图片整体做仿射变换，会使得只有特征点所在的区域仿射结果比较好，而其他的区域则不能保证。可能的解决方法是对图像先分成若干区域，对不同的区域分别做仿射变换。

② 加权方式不够好。这里我采用了简单的线性加权，能够满足基本要求，但也存在改进的空间。

③ 图片拍摄的角度问题。客观上来说，三张在不同位置拍摄的图片确实很难拼接的完美。比如在一张照片里出现的人，在另一张图片里就没有了，或者是同一个位置的树在另一张照片里由于视角的差异到了别的地方。

五、收获与总结

1.遇到的问题和解决方法

（1） MATLAB 运行太慢

由于 MATLAB 软件自身的缺陷，并且在图像处理中涉及到非常多的矩阵运算，因此在刚开始的时候我跑一遍程序可能要用到 20 分钟以上。后来我把原图调小了很多，在调试程序的时候就快了很多。

（2）特征点选取不理想

刚开始的时候为了照顾到图像的整体变换效果，我选择的特征点是平均分布在整张图片上的，但是发现仿射的效果并不好，经过多次尝试，我发现特征点应尽量选取在重合部分，这样仿射的效果才比较好。

2.收获

（1） 之前的几次小作业都只用到了 MATLAB 最基本的一些功能，通过这次大作业我对于 MATLAB 的了解和使用都提高了很多，希望能对以后的学习有所帮助；

（2） 对 MATLAB 对于图片的“理解”有了一些初步的认识，比如：

```
>> img=imread('1.jpg');
>> [m n]=size(img)

m =

        2448

n =

        9792
```

```
>> [m n k]=size(img)

m =

        2448

n =

        3264

k =

     3
```

说明对于不同的图像处理要求，MATLAB 可以把图片当成是二维矩阵，也可以当成是三维矩阵；

（3）在这次大作业中，我用到了一些数值分析中学到的插值知识，对之前的知识是一个巩固，也是不同学科之间的相互补充。